--

[gulp 1](#_Toc22487)

[基础教程 2](#_Toc25069)

[www 2](#_Toc5834)

[安装 2](#_Toc384)

[使用 2](#_Toc23911)

[API 2](#_Toc3530)

[src 2](#_Toc10186)

[dest 3](#_Toc12795)

[task 3](#_Toc15818)

[watch 4](#_Toc17003)

[run 4](#_Toc1737)

[小结 4](#_Toc9118)

[常见插件 5](#_Toc11586)

[gulp-clean 5](#_Toc22043)

[重命名gulp-rename 5](#_Toc6068)

[uglify 5](#_Toc13874)

[gulp-clean-css 5](#_Toc10698)

[url-md5 6](#_Toc14073)

[html文件压缩 6](#_Toc6139)

[js代码检查 6](#_Toc23028)

[文件合并 6](#_Toc10489)

[less和sass的编译 6](#_Toc29006)

[图片压缩 6](#_Toc23448)

[gulp-livereload 6](#_Toc26764)

[gulp-connect 6](#_Toc23848)

[browserSync 7](#_Toc28701)

[require-dir 7](#_Toc23374)

[gulp-tap  7](#_Toc2622)

[run-sequence  7](#_Toc21975)

[使用gulp-plumber来捕获处理任务中的错误 7](#_Toc11260)

[gulp-changed 8](#_Toc30766)

[del 8](#_Toc31588)

[自动加载插件gulp-load-plugins 8](#_Toc32589)

[YUI Compressor 8](#_Toc15395)

[使用案例 9](#_Toc6240)

gulp

|  |  |
| --- | --- |
|  | gulp是工具链、构建工具，可以配合各种插件做js压缩，css压缩，less编译 替代手工实现自动化工作  1.构建工具  2.自动化  3.提高效率用  webpack是文件打包工具，可以把项目的各种js文、css文件等打包合并成一个或多个文件，主要用于模块化方案，预编译模块的方案  1.打包工具  2.模块化识别  3.编译模块代码方案用  所以定义和用法上来说 都不是一种东西，无可比性 ，更不冲突！【当然，也有相似的功能，比如合并，区分，但各有各的优势】  Gulp：  顾名思义：工程自动化，解除手动繁琐；  说到 自动化 ，也就是常说的前端构建工具，因为构建工具就是用来让我们不再做机械重复的事情，解放我们的双手和节省时间；  > 能够优化前端工作流程，大大提高效率；  > 比如自动刷新页面、雪碧图、压缩CSS、js、编译less、检查语法等  > 简单来说，就是使用Gulp，然后配置你需要的插件，就可以完成之前需要手工做的事情  Webpack：解释图【来自网络和官网】  IMG_258  Gulp 和 browserify / webpack 不是一回事  Gulp应该和Grunt比较，他们的区别我就不说了，说说用处吧。Gulp / Grunt 是一种工具，能够优化前端工作流程。比如自动刷新页面、combo、压缩css、js、编译less等等。简单来说，就是使用Gulp/Grunt，然后配置你需要的插件，就可以把以前需要手工做的事情让它帮你做了。  说到 browserify / webpack ，那还要说到 seajs / requirejs 。这四个都是JS模块化的方案。其中seajs / require 是一种类型，browserify / webpack 是另一种类型。  seajs / require : 是一种在线"编译" 模块的方案，相当于在页面上加载一个 CMD/AMD 解释器。这样浏览器就认识了 define、exports、module 这些东西。也就实现了模块化。  browserify / webpack : 是一个预编译模块的方案，相比于上面 ，这个方案更加智能。没用过browserify，这里以webpack为例。首先，它是预编译的，不需要在浏览器中加载解释器。另外，你在本地直接写JS，不管是 AMD / CMD / ES6 风格的模块化，它都能认识，并且编译成浏览器认识的JS。这样就知道，Gulp是一个工具，而webpack等等是模块化方案。Gulp也可以配置seajs、requirejs甚至webpack的插件。 |
|  |  |
|  |  |
|  |  |
|  | - node , 是javascript语言的环境和平台， - npm , bower 是一类，包管理， - webpack , browserify , 是一类，javascript预编译模块的方案， - requirejs , seajs , 是一类, 基于commonjs，amd，cmd，umd 之类的模块类包加载方案的框架， - grunt , gulp , 前端工具，合并、压缩、编译 sass/less，browser 自动载入资源， - react , angular , vue , backbone , 是一类，mvc , mvvm , mvp 之类的前端框架， - jquery , zepto , prototype , 是一类，前端 DOM , BOM 类库 ， - ext , yui , kissy , dojo , 是一类，前端应用组件， - underscore , 函数式编程库。  IMG_256 |
|  | gulp官方网址：[http://gulpjs.com](http://gulpjs.com/" \t "http://www.ydcss.com/archives/_blank)  gulp插件地址：[http://gulpjs.com/plugins](http://gulpjs.com/plugins" \t "http://www.ydcss.com/archives/_blank)  gulp 官方API：[https://github.com/gulpjs/gulp/blob/master/docs/API.md](https://github.com/gulpjs/gulp/blob/master/docs/API.md" \t "http://www.ydcss.com/archives/_blank) |
|  | gulp教程  <http://www.ydcss.com/>  gulp详细入门教程  <http://www.ydcss.com/archives/18>  gulp详细入门教程---构建工具  <http://www.ydcss.com/archives/category/%E6%9E%84%E5%BB%BA%E5%B7%A5%E5%85%B7>  教程索引：  [gulp教程之gulp中文API](http://www.ydcss.com/archives/424" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-less](http://www.ydcss.com/archives/34" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-uglify](http://www.ydcss.com/archives/54" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-concat](http://www.ydcss.com/archives/83" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-jshint](http://www.ydcss.com/archives/92" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-htmlmin](http://www.ydcss.com/archives/20" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-imagemin](http://www.ydcss.com/archives/26" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-minify-css](http://www.ydcss.com/archives/41" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-rev-append](http://www.ydcss.com/archives/49" \t "http://www.ydcss.com/archives/_blank)  [gulp教程之gulp-autoprefixer](http://www.ydcss.com/archives/94" \t "http://www.ydcss.com/archives/_blank)  <http://www.cnblogs.com/JreeyQi/p/4900864.html>  [Gulp压缩前端CS,JS,图片文件](http://www.cnblogs.com/JreeyQi/p/4900864.html)  Gulp 基于Node.js的前端构建工具，Gulp有许多插件（[这里是插件](http://gulpjs.com/plugins/" \t "http://www.cnblogs.com/JreeyQi/p/_blank)），使用Gulp可以实现前端代码的编译（sass、less）、压缩、图片的压缩等，当然主要是前端工程化，不过我目前只是需要压缩文件就，至于Grunt，相比于Gulp太慢了，[Gulp API](https://github.com/gulpjs/gulp/blob/master/docs/API.md" \t "http://www.cnblogs.com/JreeyQi/p/_blank)；  自动添加css前缀（[gulp-autoprefixer](http://github.com/Metrime/gulp-autoprefixer)）  压缩css（[gulp-minify-css](http://github.com/jonathanepollack/gulp-minify-css)）gulp-clean-css  js代码校验（[gulp-jshint](http://github.com/spenceralger/gulp-jshint)）  合并js文件（[gulp-concat](http://github.com/wearefractal/gulp-concat)）  压缩js代码（[gulp-uglify](http://github.com/terinjokes/gulp-uglify)）  压缩图片（[gulp-imagemin](http://github.com/sindresorhus/gulp-imagemin)）  自动刷新页面（[gulp-livereload](http://github.com/vohof/gulp-livereload)）  图片缓存，只有图片替换了才压缩（[gulp-cache](http://github.com/jgable/gulp-cache)）  更改提醒（[gulp-notify](http://github.com/mikaelbr/gulp-notify)）  清除文件（[del](http://www.npmjs.org/package/del)）  windows下gulp练习  <http://www.qdfuns.com/notes/13600/18d5bac206f8fd7ad34bffb8e5fed98b.html>  gulp资料收集  <https://github.com/Platform-CUF/use-gulp>  [gulp的几个重要的API](https://segmentfault.com/a/1190000004006524)  <https://segmentfault.com/a/1190000004006524>  gulp系列 1-6  <http://www.cnblogs.com/1wen/category/697844.html>  gulp系列教程  <http://cw.hubwiz.com/card/c/562089cb1bc20c980538e25b/1/1/1/> |
|  | Browsersync API  <https://browsersync.io/docs/api>  <http://www.browsersync.cn/docs/command-line/>  Browsersync + Gulp.js  <https://browsersync.io/docs/gulp/>  [BrowserSync，迅捷从免F5开始](https://segmentfault.com/a/1190000002607627)  <https://segmentfault.com/a/1190000002607627>  基于 Express+Gulp+BrowserSync 搭建一套高性能的前端开发环境  <http://www.tuicool.com/articles/IfaIZbF> |
|  | [gulp-connect浏览器自动刷新](http://www.cnblogs.com/anywing/p/5311061.html)  <http://www.cnblogs.com/anywing/p/5311061.html>  gulp教程之gulp-livereload  <http://www.ydcss.com/archives/702>  Gulp.js-livereload 不用F5了，实时自动刷新页面来开发  <https://cnodejs.org/topic/53427d16dc556e3b3901861e>  <https://www.kisnows.com/2015/11/02/dev-environment-Express-Gulp-BrowserSync/>  使用Gulp构建本地开发Web服务器Gulp + Gulp-connect +watch + livereload  <http://www.tuicool.com/articles/qUvyEj>  Match files using the patterns the shell uses, like stars and stuff.  <https://github.com/isaacs/node-glob>  [介绍一种基于gulp对seajs的模块做合并压缩的方式](http://www.cnblogs.com/lyzg/p/5581961.html)  <http://www.cnblogs.com/lyzg/p/5581961.html>  [gulp前端自动化构建工具（一二三四）](http://blog.csdn.net/franks_t_d/article/details/52767911)  <http://blog.csdn.net/franks_t_d/article/category/6434079>  [gulp前端自动化构建工具（二）：gulp插件/gulp模块介绍](http://blog.csdn.net/franks_t_d/article/details/52763310)  <http://blog.csdn.net/franks_t_d/article/details/52763310>  小插件  <https://github.com/xieranmaya/gulp-showdown>  F:\practice\小demo\gulp-showdown.js  想用markdown编写带索引的单页结构HTML文档的话，可以试试天书。  <https://github.com/tianmajs/tianshu> |
|  |  |
|  | gulp是前端开发过程中对代码进行构建的工具，是自动化项目的构建利器；她不仅能对网站资源进行优化，而且在开发过程中很多重复的任务能够使用正确的工具自动完成；使用她，我们不仅可以很愉快的编写代码，而且大大提高我们的工作效率。  gulp是基于Nodejs的自动任务运行器， 她能自动化地完成 javascript/coffee/sass/less/html/image/css 等文件的的测试、检查、合并、压缩、格式化、浏览器自动刷新、部署文件生成，并监听文件在改动后重复指定的这些步骤。在实现上，她借鉴了Unix操作系统的管道（pipe）思想，前一级的输出，直接变成后一级的输入，使得在操作上非常简单。通过本文，我们将学习如何使用Gulp来改变开发流程，从而使开发更加快速高效。  gulp 和 grunt 非常类似，但相比于 grunt 的频繁 IO 操作，gulp 的流操作，能更快地更便捷地完成构建工作。  [gulpjs](http://gulpjs.com/)是一个前端构建工具，与[gruntjs](http://gruntjs.com/)相比，gulpjs无需写一大堆繁杂的配置参数，API也非常简单，学习起来很容易，而且gulpjs使用的是nodejs中[stream](http://nodejs.org/api/stream.html)来读取和操作数据，其速度更快。 |

基础教程

|  |  |
| --- | --- |
| www | [前端构建工具gulpjs的使用介绍及技巧](http://www.cnblogs.com/2050/p/4198792.html) (安装 使用 API 插件)  <http://www.cnblogs.com/2050/p/4198792.html>  [gulp的安装](http://www.cnblogs.com/2050/p/4198792.html" \l "part1)  [开始使用gulp](http://www.cnblogs.com/2050/p/4198792.html" \l "part2)  [gulp的API介绍](http://www.cnblogs.com/2050/p/4198792.html" \l "part3)  [一些常用的gulp插件](http://www.cnblogs.com/2050/p/4198792.html" \l "part4) |
| 安装 | 1、gulp的安装  首先确保你已经正确安装了nodejs环境。然后以全局方式安装gulp：  npm install -g gulp  全局安装gulp后，还需要在每个要使用gulp的项目中都单独安装一次。把目录切换到你的项目文件夹中，然后在命令行中执行：  npm install gulp  如果想在安装的时候把gulp写进项目package.json文件的依赖中，则可以加上--save-dev：  npm install --save-dev gulp  这样就完成了gulp的安装。至于为什么在全局安装gulp后，还需要在项目中本地安装一次，有兴趣的可以看下stackoverflow上有人做出的回答：[why-do-we-need-to-install-gulp-globally-and-locally](http://stackoverflow.com/questions/22115400/why-do-we-need-to-install-gulp-globally-and-locally)、[what-is-the-point-of-double-install-in-gulp](http://stackoverflow.com/questions/25713618/what-is-the-point-of-double-install-in-gulp)。大体就是为了版本的灵活性，但如果没理解那也不必太去纠结这个问题，只需要知道通常我们是要这样做就行了。 |
| 使用 | 2、开始使用gulp  2.1 建立gulpfile.js文件  就像gruntjs需要一个Gruntfile.js文件一样，gulp也需要一个文件作为它的主文件，在gulp中这个文件叫做gulpfile.js。新建一个文件名为gulpfile.js的文件，然后放到你的项目目录中。之后要做的事情就是在gulpfile.js文件中定义我们的任务了。下面是一个最简单的gulpfile.js文件内容示例，它定义了一个默认的任务。  var gulp = require('gulp');gulp.task('default',function(){  console.log('hello world');  });  此时我们的目录结构是这样子的：  ├── gulpfile.js ├── node\_modules │ └── gulp └── package.json  2.2 运行gulp任务  要运行gulp任务，只需切换到存放gulpfile.js文件的目录(windows平台请使用cmd或者Power Shell等工具)，然后在命令行中执行gulp命令就行了，gulp后面可以加上要执行的任务名，例如gulp task1，如果没有指定任务名，则会执行任务名为default的默认任务。 |
| API | 仅仅需要知道5个gulp命令：  gulp.task(name, fn)这个你应经见过了  gulp.run(tasks...)尽可能多的并行运行多个task  gulp.watch(glob, fn)当glob内容发生改变时，执行fn  gulp.src(glob)返回一个可读的stream  gulp.dest(glob)返回一个可写的stream |
| src | 3.1 gulp.src()  在介绍这个API之前我们首先来说一下Grunt.js和Gulp.js工作方式的一个区别。Grunt主要是以文件为媒介来运行它的工作流的，比如在Grunt中执行完一项任务后，会把结果写入到一个临时文件中，然后可以在这个临时文件内容的基础上执行其它任务，执行完成后又把结果写入到临时文件中，然后又以这个为基础继续执行其它任务...就这样反复下去。而在Gulp中，使用的是Nodejs中的[stream](http://nodejs.org/api/stream.html)(流)，首先获取到需要的stream，然后可以通过stream的pipe()方法把流导入到你想要的地方，比如Gulp的插件中，经过插件处理后的流又可以继续导入到其他插件中，当然也可以把流写入到文件中。所以Gulp是以stream为媒介的，它不需要频繁的生成临时文件，这也是Gulp的速度比Grunt快的一个原因。再回到正题上来，gulp.src()方法正是用来获取流的，但要注意这个流里的内容不是原始的文件流，而是一个虚拟文件对象流([Vinyl files](https://github.com/wearefractal/vinyl-fs))，这个虚拟文件对象中存储着原始文件的路径、文件名、内容等信息，这个我们暂时不用去深入理解，你只需简单的理解可以用这个方法来读取你需要操作的文件就行了。其语法为：  gulp.src(globs[, options]) |
|  | globs参数是文件匹配模式(类似正则表达式)，用来匹配文件路径(包括文件名)，当然这里也可以直接指定某个具体的文件路径。当有多个匹配模式时，该参数可以为一个数组;类型为String或 Array。  [globs的匹配规则请看这个链接](http://www.bruceit.com/?p=204" \t "https://segmentfault.com/a/_blank)：http://cw.hubwiz.com/card/c/562089cb1bc20c980538e25b/1/2/2/  当有多种匹配模式时可以使用数组：gulp.src(['js/\*.js','css/\*.css','\*.html'])  options为可选参数。以下为options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 ， 设置输出路径以某个路径的某个组成部分为基础向后拼接。 |
|  | 下面我们重点说说Gulp用到的glob的匹配规则以及一些文件匹配技巧。Gulp内部使用了[node-glob](https://github.com/isaacs/node-glob)模块来实现其文件匹配功能。我们可以使用下面这些特殊的字符来匹配我们想要的文件：  \* 匹配文件路径中的0个或多个字符，但不会匹配路径分隔符，除非路径分隔符出现在末尾  \*\* 匹配路径中的0个或多个目录及其子目录,需要单独出现，即它左右不能有其他东西了。如果出现在末尾，也能匹配文件。  ? 匹配文件路径中的一个字符(不会匹配路径分隔符)  [...] 匹配方括号中出现的字符中的任意一个，当方括号中第一个字符为^或!时，则表示不匹配方括号中出现的其他字符中的任意一个，类似js正则表达式中的用法  !(pattern|pattern|pattern) 匹配任何与括号中给定的任一模式都不匹配的  ?(pattern|pattern|pattern) 匹配括号中给定的任一模式0次或1次，类似于js正则中的(pattern|pattern|pattern)?  +(pattern|pattern|pattern) 匹配括号中给定的任一模式至少1次，类似于js正则中的(pattern|pattern|pattern)+  \*(pattern|pattern|pattern) 匹配括号中给定的任一模式0次或多次，类似于js正则中的(pattern|pattern|pattern)\*  @(pattern|pattern|pattern) 匹配括号中给定的任一模式1次，类似于js正则中的(pattern|pattern|pattern)  下面以一系列例子来加深理解  \* 能匹配 a.js,x.y,abc,abc/,但不能匹配a/b.js  \*.\* 能匹配 a.js,style.css,a.b,x.y  \*/\*/\*.js 能匹配 a/b/c.js,x/y/z.js,不能匹配a/b.js,a/b/c/d.js  \*\* 能匹配 abc,a/b.js,a/b/c.js,x/y/z,x/y/z/a.b,\*\*能用来匹配所有的目录和文件  \*\*/\*.js 能匹配 foo.js,a/foo.js,a/b/foo.js,a/b/c/foo.js  a/\*\*/z 能匹配 a/z,a/b/z,a/b/c/z,a/d/g/h/j/k/z  a/\*\*b/z 能匹配 a/b/z,a/sb/z,但不能匹配a/x/sb/z,因为只有单\*\*单独出现才能匹配多级目录  ?.js 能匹配 a.js,b.js,c.js  a?? 能匹配 a.b,abc,但不能匹配ab/,因为它不会匹配路径分隔符  [xyz].js 只能匹配 x.js,y.js,z.js,不会匹配xy.js,xyz.js等,整个中括号只代表一个字符  [^xyz].js 能匹配 a.js,b.js,c.js等,不能匹配x.js,y.js,z.js |
|  | 当有多种匹配模式时可以使用数组  //使用数组的方式来匹配多种文件  gulp.src(['js/\*.js','css/\*.css','\*.html'])  使用数组的方式还有一个好处就是可以很方便的使用排除模式，在数组中的单个匹配模式前加上!即是排除模式，它会在匹配的结果中排除这个匹配，要注意一点的是不能在数组中的第一个元素中使用排除模式  gulp.src([\*.js,'!b\*.js']) //匹配所有js文件，但排除掉以b开头的js文件  gulp.src(['!b\*.js',\*.js]) //不会排除任何文件，因为排除模式不能出现在数组的第一个元素中 |
|  | 此外，还可以使用展开模式。展开模式以花括号作为定界符，根据它里面的内容，会展开为多个模式，最后匹配的结果为所有展开的模式相加起来得到的结果。展开的例子如下：  a{b,c}d 会展开为 abd,acd  a{b,}c 会展开为 abc,ac  a{0..3}d 会展开为 a0d,a1d,a2d,a3d  a{b,c{d,e}f}g 会展开为 abg,acdfg,acefg  a{b,c}d{e,f}g 会展开为 abdeg,acdeg,abdeg,abdfg |
| dest | gulp.dest()方法是用来写文件的，其语法为：  gulp.dest(path[,options])  > path为写入文件的路径  > options为一个可选的参数对象，通常我们不需要用到  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。 |
|  | 要想使用好gulp.dest()这个方法，就要理解给它传入的路径参数与最终生成的文件的关系。  gulp的使用流程一般是这样子的：首先通过gulp.src()方法获取到我们想要处理的文件流，然后把文件流通过pipe方法导入到gulp的插件中，最后把经过插件处理后的流再通过pipe方法导入到gulp.dest()中，gulp.dest()方法则把流中的内容写入到文件中，这里首先需要弄清楚的一点是，我们给gulp.dest()传入的路径参数，只能用来指定要生成的文件的目录，而不能指定生成文件的文件名，它生成文件的文件名使用的是导入到它的文件流自身的文件名，所以生成的文件名是由导入到它的文件流决定的，即使我们给它传入一个带有文件名的路径参数，然后它也会把这个文件名当做是目录名，例如：  var gulp = require('gulp');  gulp.src('script/jquery.js')  .pipe(gulp.dest('dist/foo.js'));  //最终生成的文件路径为 dist/foo.js/jquery.js,而不是dist/foo.js  > 要想改变文件名，可以使用插件[gulp-rename](https://www.npmjs.com/packages/gulp-rename)  > 用gulp.dest()把文件流写入文件后，文件流仍然可以继续使用。 |
|  | 下面说说生成的文件路径与我们给gulp.dest()方法传入的路径参数之间的关系。 **/\* \* gulp.dest(path)生成的文件路径是我们传入的path参数后面再加上gulp.src()中有通配符开始出现的那部分路径。 \* \*/** //没有通配符出现的情况,最后生成的文件路径为 dist/avalon.js **gulp**.src('script/avalon/avalon.js')  .pipe(**gulp**.dest('dist'));  //有通配符开始出现的那部分路径为 \*\*/underscore.js //假设匹配到的文件为script/util/underscore.js //则最后生成的文件路径为 dist/util/underscore.js **gulp**.src('script/\*\*/underscore.js')  .pipe(**gulp**.dest('dist'));  //例如：有通配符开始出现的那部分路径为 \*\*!/!\*.js, 最后生成的文件路径为 dist/!\*\*!/!\*.js //如果 \*\*!/!\*.js 匹配到的文件为 jquery/jquery.js ,则生成的文件路径为 dist/jquery/jquery.js **gulp**.src('script/\*\*/\*.js')  .pipe(**gulp**.dest('dist'));  //有通配符出现的那部分路径为 \* //假设匹配到的文件为script/zepto.js //则最后生成的文件路径为 dist/zepto.js **gulp**.src('script/\*')  .pipe(**gulp**.dest('dist'));  **/\* \* 通过指定gulp.src()方法配置参数中的base属性，我们可以更灵活的来改变gulp.dest()生成的文件路径。 \* 当我们没有在gulp.src()方法中配置base属性时，base的默认值为通配符开始出现之前那部分路径， \* \* 上面我们说的gulp.dest()所生成的文件路径的规则，其实也可以理解成： \* 用我们给gulp.dest()传入的路径替换掉gulp.src()中的base路径，最终得到生成文件的路径。 \* \*/** // 此时base的值为 app/src // 设该模式匹配到了文件 app/src/css/normal.css // 用dist替换掉base路径，最终得到 dist/css/normal.css // 所以改变base路径后，gulp.dest()生成的文件路径也会改变 **gulp**.src('app/src/\*\*/\*.css')  .pipe(**gulp**.dest('dist'));  //没有配置base参数，此时默认的base路径为script/lib //假设匹配到的文件为script/lib/jquery.js //生成的文件路径为 build/jquery.js **gulp**.src("script/lib/\*.js")  .pipe(**gulp**.dest('build'))  //配置了base参数，此时base路径为script //假设匹配到的文件为script/lib/jquery.js //此时生成的文件路径为 build/lib/jquery.js **gulp**.src("script/lib/\*.js", {base: 'script'})  .pipe(**gulp**.dest('build'))  **gulp**.src("./src/main/text/hello.txt", { base: process.cwd() })  //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' 现在 `base` 的值为 `client/js/`  .pipe(minify())  .pipe(gulp.dest('build')); // 写入 'build/somedir/somefile.js' 将`client/js/`替换为build  gulp.src('client/js/\*\*/\*.js', { base: 'client' }) // base 的值为 'client'  .pipe(minify())  .pipe(gulp.dest('build')); // 写入 'build/js/somedir/somefile.js' 将`client`替换为build |
| task | gulp.task方法用来定义任务，内部使用的是[Orchestrator](https://github.com/robrich/orchestrator)，其语法为：  gulp.task(name[, deps], fn)  name 为任务名  deps 是当前定义的任务需要依赖的其他任务，为一个数组。当前定义的任务会在所有依赖的任务执行完毕后才开始执行。如果没有依赖，则可省略这个参数  fn 为任务函数，我们把任务要执行的代码都写在里面。该参数也是可选的。 |
|  | gulp.task()这个API没什么好讲的，但需要知道执行多个任务时怎么来控制任务执行的顺序。  gulp中执行多个任务，可以通过任务依赖来实现。例如我想要执行one,two,three这三个任务，那我们就可以定义一个空的任务，然后把那三个任务当做这个空的任务的依赖就行了：  //只要执行default任务，就相当于把one,two,three这三个任务执行了  gulp.task('default',['one','two','three']);  如果任务相互之间没有依赖，任务会按你书写的顺序来执行，如果有依赖的话则会先执行依赖的任务。  但是如果某个任务所依赖的任务是异步的，就要注意了，gulp并不会等待那个所依赖的异步任务完成，而是会接着执行后续的任务。 |
|  | 例如：  gulp.task('one', function () {  //one是一个异步执行的任务  setTimeout(function () {  console.log('one is done');  }, 5000);  });  //two任务虽然依赖于one任务,但并不会等到one任务中的异步操作完成后再执行  gulp.task('two', ['one'], function () {  console.log('two is done');  });  gulp.run("two");  上面的例子中我们执行two任务时，会先执行one任务，但不会去等待one任务中的异步操作完成后再执行two任务，而是紧接着执行two任务。所以two任务会在one任务中的异步操作完成之前就执行了。 |
|  | 那如果我们想等待异步任务中的异步操作完成后再执行后续的任务，该怎么做呢？有三种方法可以实现：  第一：在异步操作完成后执行一个回调函数来通知gulp这个异步任务已经完成,这个回调函数就是任务函数的第一个参数。  gulp.task('one',function(cb){ //cb为任务函数提供的回调，用来通知任务已经完成  //one是一个异步执行的任务  setTimeout(function(){  console.log('one is done');  cb(); //执行回调，表示这个异步任务已经完成  },5000);  });  //这时two任务会在one任务中的异步操作完成后再执行  gulp.task('two',['one'],function(){//定义一个有依赖的任务  console.log('two is done');  });  第二：定义任务时返回一个流对象。适用于任务就是操作gulp.src获取到的流的情况。  gulp.task('one',function(cb){  var stream = gulp.src('client/\*\*/\*.js')  .pipe(dosomething()) //dosomething()中有某些异步操作  .pipe(gulp.dest('build'));  return stream;  });  gulp.task('two',['one'],function(){  console.log('two is done');  });  第三：返回一个promise对象，例如  var Q = require('q'); //一个著名的异步处理的库 https://github.com/kriskowal/q  gulp.task('one',function(cb){  var deferred = Q.defer();  // 做一些异步操作  setTimeout(function() {  deferred.resolve();  }, 5000);  return deferred.promise;  });  gulp.task('two',['one'],function(){  console.log('two is done');  });  gulp.task('default', () => {  return new Promise(resolve => {  dev = false;  runSequence(['clean', 'wiredep'], 'build', resolve);  });  }); |
| watch | gulp.watch()用来监视文件的变化，当文件发生变化后，我们可以利用它来执行相应的任务，例如文件压缩等。其语法为  gulp.watch(glob[, opts], tasks)  glob 为要监视的文件匹配模式，规则和用法与gulp.src()方法中的glob相同。  opts 为一个可选的配置对象，通常不需要用到  tasks 为文件变化后要执行的任务，为一个数组 |
| gulp.watch(glob[, opts, cb])  glob和opts参数与第一种用法相同  cb参数为一个函数。每当监视的文件发生变化时，就会调用这个函数,并且会给它传入一个对象，该对象包含了文件变化的一些信息，type属性为变化的类型，可以是added,changed,deleted；path属性为发生变化的文件的路径 |
|  | gulp.task('uglify', **function** () {  //do something }); gulp.task('reload', **function** () {  //do something });  **/\*  gulp.watch(glob [, opts], tasks) or gulp.watch(glob [, opts, cb])  说明：watch方法是用于监听文件变化，文件一修改就会执行指定的任务；  glob：  需要处理的源文件匹配符路径。类型(必填)：String or StringArray；  opts：  类型(可选)：Object 具体参看https://github.com/shama/gaze；  tasks：  类型(必填)：StringArray 需要执行的任务的名称数组；  cb(event)：  类型(可选)：Function 每个文件变化执行的回调函数；  \*/** gulp.task('watch1', **function** () {  **var** watcher = gulp.watch('js/\*\*/\*.js', ['uglify', 'reload']);  watcher.on('change', **function** (event) {  console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');  }); });  gulp.task('watch2', **function** () {  gulp.watch('js/\*\*/\*.js', **function** (event) {  console.log(event.type); //变化类型 added为新增,deleted为删除，changed为改变  console.log(event.path); //变化的文件的路径  }); }); |
| run | gulp.run()表示要执行的任务。可能会使用单个参数的形式传递多个任务。如下代码：  gulp.task('end',function(){  gulp.run('task1','task3','task2');  });  注意：任务是尽可能多的并行执行的，并且可能不会按照指定的顺序运行。 |

小结

|  |  |
| --- | --- |
|  | [gulp使用小结(一)](http://www.cnblogs.com/Darren_code/p/gulp.html)  http://www.cnblogs.com/Darren\_code/p/gulp.html  必备插件  【必备】不是指每次项目都得用，而是指最基础的。你可以不用，但是得知道的那些gulp插件  以下插件的方向都各自有不少相似的插件，所以不必过多纠结，用的开心就行  [gulp-htmlmin](https://www.npmjs.com/package/gulp-htmlmin) 看到名字就能知道，这个插件是用来压缩 HTML。PS：注释啥的都可以去掉哦，看文档get更多技能哈  [gulp-imagemin](https://www.npmjs.com/package/gulp-imagemin) 除了能压缩常见的图片格式，还能压缩 SVG，叼叼的~  [gulp-clean-css](https://www.npmjs.com/package/gulp-minify-css) 压缩 CSS。我原本推荐的是 [gulp-minify-css](https://www.npmjs.com/package/gulp-minify-css)，结果其首页中已建议改用gulp-clean-css...  [gulp-uglify](https://www.npmjs.com/package/gulp-uglify) 专业压缩 Javascript  [gulp-concat](https://www.npmjs.com/package/gulp-concat) 上面几个都是压缩，这插件是管合并的...恭喜，“减少网络请求”的成就达成:)  [gulp-autoprefixer](https://www.npmjs.com/package/gulp-autoprefixer) 给 CSS 增加前缀。解决某些CSS属性不是标准属性，有各种浏览器前缀的情况，灰常有用  [gulp-rename](https://www.npmjs.com/package/gulp-rename) 修改文件名称。比如有时我们需要把app.js改成app.min.js，瞬间高级了  [gulp-util](https://www.npmjs.com/package/gulp-util) 最基础的工具，但俺只用来打日志...  常用插件  相对于必备插件，常用插件都是些【有用】但使用场景和频率没那么高的好插件  [run-sequence](https://www.npmjs.com/package/run-sequence) gulp 的 task 都是并行(异步)执行，如果遇见需要串行的场景，那么这个插件就是必备了。偶的使用场景是：处理(压缩、合并等等) CSS/JS、再[gulp-rev](https://www.npmjs.com/package/gulp-rev)、再上传 CDN；然后使用 CDN的地址替换 HTML 中的 CSS/JS 地址，再压缩 HTML。那么替换 HTML 这步须在之前的工作处理完后再执行。 \*\* 最后要说，gulp4.0发布后，不需要RS也可以搞定串行任务了 \*\*  [del](https://www.npmjs.com/package/del) / [gulp-clean](https://www.npmjs.com/package/gulp-clean) 删除。俺的使用场景是：JS/CSS 文件都会在压缩后使用[gulp-rev](https://www.npmjs.com/package/gulp-rev)，即文件名被hash，然后再上传到 CDN，最后俺再使用 删除插件 把本地压缩后的文件删除掉，不用多余保存。  [gulp-rev](https://www.npmjs.com/package/gulp-rev) 把静态文件名改成hash的形式。  [gulp-rev-replace](https://www.npmjs.com/package/gulp-rev-replace) 配合 [gulp-rev](https://www.npmjs.com/package/gulp-rev) 使用，拿到生成的 manifest。json 后替换对应的文件名称。  [gulp-rev-collector](https://www.npmjs.com/package/gulp-rev-collector) 到线上环境前，我会用来配合[gulp-rev](https://www.npmjs.com/package/gulp-rev)使用，替换 HTML 中的路径  [gulp-rev-append](https://www.npmjs.com/package/gulp-rev-append) 给页面引用的静态文件增加hash后缀，避免被浏览器缓存...当然，如果是使用 CDN，这个套路就不行了  [gulp-connect](https://www.npmjs.com/package/gulp-connect) / [gulp-livereload](https://www.npmjs.com/package/gulp-livereload) LiveReload 的俩款插件都值得拥有，不过都各稍有学习成本，看看文档就明白鸟  [gulp-sourcemaps](https://www.npmjs.com/package/gulp-sourcemaps) 处理 JavaScript 时生成 SourceMap；如果你不了解 SourceMap，可以看看这篇阮一峰大神的[《Source Map 详解》](http://www.ruanyifeng.com/blog/2013/01/javascript_source_map.html)  [gulp-load-plugins](https://www.npmjs.com/package/gulp-load-plugins) 帮忙偷懒用的，可以帮我们加载插件，不用require或者import...当然，俺个人感觉用了这个插件后，阅读gulpfile.js的可读性差了，鱼和熊掌不可兼得:(  [gulp-jshint](https://www.npmjs.com/package/gulp-jshint) JavaScript 代码校验  [gulp-sass](https://www.npmjs.com/package/gulp-sass) / [gulp-less](https://www.npmjs.com/package/gulp-less) 写 CSS 的同学都懂哈  进阶插件  必备插件和常用插件都有一个特点，就是下载数都不小。虽然有少部分插件下载数相对不多，但同样值得拥有或了解  [babel](https://babeljs.io/) JS 语法新特性用起来。这个插件可以让我们用新的 标准/特性/提案 写 JavaScript 代码，然后再向下 转换编译，最终生成随处可用的 JavaScript 代码。更通俗的说话就是：可以用新的规范写代码，经过 babel 编译后生成没有兼容问题的代码。  [gulp-flatten](https://www.npmjs.com/package/gulp-flatten) 移动指定文件，不想压缩或者合并的时候，直接用这个插件把对应文件移动到指定文件夹。俺偶尔在内部项目会偷懒用上，图方便:)  [gulp-coffee](https://www.npmjs.com/package/gulp-coffee) CoffeeScript 值得去了解  [gulp-markdown-pdf](https://www.npmjs.com/package/gulp-markdown-pdf) 把 Markdown 编译为 PDF  [gulp-markdown](https://www.npmjs.com/package/gulp-markdown) 写手的福音，可以把 Markdown 转成 HTML  [gulp-html2md](https://www.npmjs.com/package/gulp-html2md) 把 HTML 编译为 Markdown  [gulp-tinypng](https://www.npmjs.com/package/gulp-tinypng) 超屌的图片压缩工具，使用 Tinypng 引擎。PS：因为 Tinypng 免费帐号有月限制，所以使用使需注意。  [sprity](https://www.npmjs.com/package/sprity) 生成雪碧图。稍有点学习成本，仔细阅读文档即可。  [gulp-if](https://www.npmjs.com/package/gulp-if) 可以在 pipe 里面写点逻辑了，屌不屌。举例：比如处理 ./pub/\*.js，如果文件名称是 xxx.js，那么不处理；更可以用来区分当前是开发环境还是生产环境。  [gulp-file-include](https://www.npmjs.com/package/gulp-file-include) 俺搞内部项目的时候会用到，让 HTML 组件化的第一小步  [gulp-git](https://www.npmjs.com/package/gulp-git) 直接在 Build 时把代码都提交到 git上了...特么劳资懒起来连我自己都害怕  [gulp-qiniu](https://www.npmjs.com/package/gulp-qiniu) 用于把指定文件上传至七牛的指定路径下(PS:首先，你得有自己的七牛账号和空间)  [gulp-notify](https://www.npmjs.com/package/gulp-notify) 在控制台中加入文字描述，build 的时候更高级有木有。当然，当需要的时候把错误信息也展示出来会很有帮助。更高级的功能就需要你查看其文档了~  [gulp-plumber](https://www.npmjs.com/package/gulp-plumber) gulp 的错误处理有点坑，假如发生错误进程就挂了。相对的解决办法不少，但是这个是我个人比较推荐的，比特么在容易出错的地方写错误监听靠谱。所以这个插件可以阻止 gulp 插件发生错误导致进程退出并输出错误日志。  任务错误中断自动重传（gulp-plumber）  好吧，“任务错误中断自动重传”是我瞎命名的。默认的 Gulp 任务在执行过程中如果出错会报错并立即停止当前工作流（如在 watch Sass编译时候恰巧 Sass代码写错了）。使用plumber 模块可以在纠正错误后继续执行任务。  自动打包并按时间重命名（gulp-zip）  一般项目完成后需要整理文件并压缩以供交付使用或进行下一阶段的开发，本模块可以实现将项目文件自动打包并按时间重命名。 |

常见插件

|  |  |  |  |
| --- | --- | --- | --- |
| 常见插件 | sass的编译（gulp-sass） less编译 （gulp-less） 重命名（gulp-rename） 自动添加css前缀（gulp-autoprefixer） 压缩css（gulp-clean-css） js代码校验（gulp-jshint） 合并js文件（gulp-concat） 压缩js代码（gulp-uglify） 压缩图片（gulp-imagemin） 自动刷新页面（gulp-livereload，谷歌浏览器亲测，谷歌浏览器需安装livereload插件） 图片缓存，只有图片替换了才压缩（gulp-cache） 更改提醒（gulp-notify） | | |
|  | gulp-if  gulp-rev-replace  del  shelljs  gulp-useref  gulp-angular-templatecache  gulp-ng-annotate  gulp-jshint  gulpIgnore  gulp-replace  gulp-processheml  gulp-cssnao  gulp-copy-rex  rimraf  clean | | |
| gulp-clean | <http://www.cnblogs.com/1wen/p/4586198.html> | | |
| npm install --save-dev gulp-clean  var gulp = require('gulp');  var clean = require('gulp-clean');  gulp.task("clean", function () {  return gulp.src("./build")  .pipe(clean());  })  //清理  gulp.task('clean', function() {  return gulp.src(['dist/css', 'dist/js', 'dist/images'], {read: false})  .pipe(clean());  });  //注意:下面这样是错误的；因为这样写，这些任务是同步的，完全可能出现边编译边删除的情况:  gulp.task('default', ['clean', 'less', 'images', 'js', 'watch']);  //所以需要配置一个异步，非常简单，加个回调[预设任务]:  gulp.task('default', ['clean'], function(){  gulp.start('less', 'images', 'js', 'watch');  }); | | |
|  |  | | |
| 重命名gulp-rename | 使用[gulp-rename](https://www.npmjs.com/packages/gulp-rename)  安装：npm install --save-dev gulp-rename  用来重命名文件流中的文件。用gulp.dest()方法写入文件时，文件名使用的是文件流中的文件名，如果要想改变文件名，那可以在之前用gulp-rename插件来改变文件流中的文件名。  gulp.task('rename', **function** () {  gulp.src('js/jquery.js')  .pipe(uglify()) //压缩  .pipe(rename('jquery.min.js')) //会将jquery.js重命名为jquery.min.js  .pipe(gulp.dest('js')); });  关于gulp-rename的更多强大的用法请参考：<https://www.npmjs.com/package/gulp-rename> | | |
| **var rename** = **require**("gulp-rename");  // rename via string  **gulp**.src("./src/main/text/hello.txt")  .pipe(**rename**("main/text/ciao/goodbye.md"))  .pipe(**gulp**.dest("./dist")); // ./dist/main/text/ciao/goodbye.md   // rename via function  **gulp**.src("./src/\*\*/hello.txt")  .pipe(**rename**(**function** (path) {  path.dirname += "/ciao";  path.basename += "-goodbye";  path.extname = ".md"  }))  .pipe(**gulp**.dest("./dist")); // ./dist/main/text/ciao/hello-goodbye.md   // rename via hash  **gulp**.src("./src/main/text/hello.txt", { base: process.cwd() })  .pipe(rename({  dirname: "main/text/ciao",  basename: "aloha",  prefix: "bonjour-",  suffix: "-hola",  extname: ".md"  }))  .pipe(gulp.dest("./dist")); // ./dist/main/text/ciao/bonjour-aloha-hola.md | | |
| uglify | [gulp-uglify](https://www.npmjs.com/packages/gulp-uglify)  git：<https://github.com/terinjokes/gulp-uglify>  安装：npm install --save-dev gulp-uglify  说明：用来压缩js文件，使用的是uglify引擎  var gulp = require('gulp'),  uglify = require("gulp-uglify");  gulp.task('minify-js', function () {  gulp.src('js/\*.js') // 要压缩的js文件  .pipe(uglify()) //使用uglify进行压缩,更多配置请参考：  .pipe(gulp.dest('dist/js')); //压缩后的路径  }); | | |
|  | var gulp = require('gulp');  var uglify = require('gulp-uglify');  var pump = require('pump');  gulp.task('compress', function (cb) {  pump([  gulp.src('lib/\*.js'),  uglify(),  gulp.dest('dist')  ],  cb  );  }); | | |
| gulp-clean-css | <https://github.com/scniro/gulp-clean-css>  说明：[gulp](http://gulpjs.com/) plugin to minify CSS, using [clean-css](https://github.com/jakubpawlowicz/clean-css)  安装：npm install gulp-clean-css --save-dev  API：cleanCSS([options], [callback])  使用：  var gulp = require('gulp'),  cleanCSS= require("gulp-clean-css"); | | |
| > options  gulp.task('minify-css', function () {  gulp.src('css/\*.css') // 要压缩的css文件  .pipe(cleanCSS({compatibility: 'ie8'})) //压缩css ie8+兼容模式  .pipe(gulp.dest('dist/css'));  });  > callback  gulp.task('minify-css', function() {  return gulp.src('styles/\*.css')  .pipe(cleanCSS({debug: true}, function(details) {  console.log(details.name + ': ' + details.stats.originalSize);  console.log(details.name + ': ' + details.stats.minifiedSize);  }))  .pipe(gulp.dest('dist'));  }); | | |
| advanced: false, | //类型：Boolean 默认：true [是否开启高级优化(合并选择器等)] | |
| compatibility: 'ie7', | //保留ie7及以下兼容写法 类型：String 默认：''or'\*' [启用兼容模式； 'ie7'：IE7兼容模式，'ie8'：IE8兼容模式，'\*'：IE9+兼容模式] | |
| keepBreaks: true, | //类型：Boolean 默认：false [是否保留换行] | |
| keepSpecialComments: '\*' | //保留所有特殊前缀：当你用autoprefixer生成的浏览器前缀，如果不加这个参数，有可能将会删除你的部分前缀 | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
| 若想保留注释，这样注释即可：  /\*!     Important comments included in minified output.  \*/ | | |
| url-md5 | 给css文件里引用url加版本号（根据引用文件的md5生产版本号），确保已本地安装gulp-make-css-url-version [cnpm install gulp-make-css-url-version --save-dev]  var cssver = require('gulp-make-css-url-version');  gulp.src('src/css/\*.css')  .pipe(cssver()) //给css文件里引用文件加版本号（文件MD5）  .pipe(cssmin())  .pipe(gulp.dest('dist/css'));  IMG_256 | | |
| html文件压缩 | 使用[gulp-minify-html](https://www.npmjs.com/packages/gulp-minify-html)  安装：npm install --save-dev gulp-minify-html  用来压缩html文件  var gulp = require('gulp'),  minifyHtml = require("gulp-minify-html");    gulp.task('minify-html', function () {  gulp.src('html/\*.html') // 要压缩的html文件  .pipe(minifyHtml()) //压缩  .pipe(gulp.dest('dist/html'));  }); | | |
| js代码检查 | 使用[gulp-jshint](https://www.npmjs.com/packages/gulp-jshint)  安装：npm install --save-dev gulp-jshint  用来检查js代码  var gulp = require('gulp'),  jshint = require("gulp-jshint");    gulp.task('jsLint', function () {  gulp.src('js/\*.js')  .pipe(jshint())  .pipe(jshint.reporter()); // 输出检查结果  }); | | |
| 文件合并 | 使用[gulp-concat](https://www.npmjs.com/packages/gulp-concat) 安装：npm install --save-dev gulp-concat 用来把多个文件合并为一个文件,我们可以用它来合并js或css文件等，这样就能减少页面的http请求数了  var gulp = require('gulp'),  concat = require("gulp-concat");    gulp.task('concat', function () {  gulp.src('js/\*.js') //要合并的文件  .pipe(concat('all.js')) // 合并匹配到的js文件并命名为 "all.js"  .pipe(gulp.dest('dist/js'));  }); | | |
| less和sass的编译 | less使用[gulp-less](https://www.npmjs.com/packages/gulp-less),安装：npm install --save-dev gulp-less  var gulp = require('gulp'),  less = require("gulp-less");    gulp.task('compile-less', function () {  gulp.src('less/\*.less')  .pipe(less())  .pipe(gulp.dest('dist/css'));  });  sass使用[gulp-sass](https://www.npmjs.com/packages/gulp-sass),安装：npm install --save-dev gulp-sass  var gulp = require('gulp'),  sass = require("gulp-sass");    gulp.task('compile-sass', function () {  gulp.src('sass/\*.sass')  .pipe(sass())  .pipe(gulp.dest('dist/css'));  }); | | |
| gulp-plumber | Briefly it replaces pipe method and removes standard onerror handler on error event, which unpipes streams on error by default.  var  plumber = require('gulp-plumber');  gulp.src('./src/\*.scss')  .pipe(plumber())  .pipe(sass())  .pipe(uglify())  .pipe(plumber.stop())  .pipe(gulp.dest('./dist'));  var  plumber = require('gulp-plumber');  var  coffee = require('gulp-coffee');  gulp.src('./src/\*.ext')  .pipe(plumber())  .pipe(coffee())  .pipe(gulp.dest('./dist')); | | |
| 图片压缩 | 可以使用[gulp-imagemin](https://github.com/sindresorhus/gulp-imagemin)插件来压缩jpg、png、gif等图片。  安装：npm install --save-dev gulp-imagemin  var gulp = require('gulp');  var imagemin = require('gulp-imagemin');  var pngquant = require('imagemin-pngquant'); //png图片压缩插件  gulp.task('default', function () {  return gulp.src('src/images/\*')  .pipe(imagemin({  progressive: true,  use: [pngquant()] //使用pngquant来压缩png图片  }))  .pipe(gulp.dest('dist'));  });  gulp-imagemin的使用比较复杂一点，而且它本身也有很多[插件](https://www.npmjs.com/browse/keyword/imageminplugin)，建议去它的[项目主页](https://github.com/sindresorhus/gulp-imagemin)看看文档 | | |
| gulp-livereload | 自动刷新:使用[gulp-livereload](https://github.com/vohof/gulp-livereload)插件，  安装:  npm install --save-dev gulp-livereload  当代码变化时，它可以帮我们自动刷新页面；该插件最好配合谷歌浏览器来使用，且要安装[livereload chrome extension](https://chrome.google.com/webstore/detail/livereload/jnihajbhpnppcggbcgedagnkighmdlei)扩展插件,不能下载的请自行FQ。  var gulp = require('gulp'),  less = require('gulp-less'),  livereload = require('gulp-livereload');  gulp.task('less', function() {  gulp.src('less/\*.less')  .pipe(less())  .pipe(gulp.dest('css'))  .pipe(livereload());  });  gulp.task('watch', function() {  livereload.listen(); //要在这里调用listen()方法  gulp.watch('less/\*.less', ['less']);  }); | | |
| gulp-connect | //创建完成后，直接访问localhost:8080即可  //定义依赖和插件  var gulp = require('gulp'),  uglify = require('gulp-uglify'),  concat = require('gulp-concat'),  rename = require('gulp-rename'),  connect = require('gulp-connect');//livereload  var jsSrc = './app/js/\*\*/\*.js';  var jsDist = 'build/js';  var htmlSrc = './app/\*\*/\*.html';  var htmlDist = 'build';  var cssSrc = './app/css/custom';  var cssDist = 'build/css';  //定义名为js的任务  gulp.task('js', function () {  gulp.src(jsSrc)  .pipe(gulp.dest(jsDist))  .pipe(connect.reload());  });  //定义html任务  gulp.task('html', function () {  gulp.src(htmlSrc)  .pipe(gulp.dest(htmlDist))  .pipe(connect.reload());  });  //定义css任务  gulp.task('css', function () {  gulp.src(cssSrc)  .pipe(gulp.dest(cssDist))  .pipe(connect.reload());  });  //定义livereload任务  gulp.task('connect', function () {  connect.server({  root: './app',  livereload: true  });  });  //定义监视任务  gulp.task('watch', function () {  gulp.watch('./app/\*\*/\*.html', ['html']);  gulp.watch('./app/js/\*\*/\*.js', ['js']);  gulp.watch('./app/css/custom/\*.css', ['css']);  });  //定义默认任务  gulp.task('default', ['js', 'html', 'css', 'watch', 'connect']); | | |
| browserSync | //创建完成后，会自动打开浏览器localhost:3000  var gulp = require('gulp'),  uglify = require('gulp-uglify'),  browserSync = require('browser-sync').create();  gulp.task('serve', function () {  browserSync.init({  server: "./app"  });  gulp.watch("./app/css/custom/\*.css").on('change', browserSync.reload);  gulp.watch("./app/\*\*/\*.html").on('change', browserSync.reload);  gulp.watch("./app/js/\*\*/\*.js").on('change', browserSync.reload);  });  gulp.task('default', ['serve']); | | |
| require-dir | [利用 gulp 的 require-dir插件管理tasks](https://segmentfault.com/q/1010000004231247)  <http://macr.ae/article/splitting-gulpfile-multiple-files.html#approach-one> | | |
| In some projects, your gulpfile will get big enough that it's worth splitting it into multiple files, with one task per file. There are a couple approaches you can take to achieve this; both are pretty simple.  Here is a fairly simple gulpfile which we will be splitting up in this article. It doesn't do anything fancy, it just deals with our JavaScript and Sass:  var gulp = require('gulp');  var concat = require('gulp-concat');  var uglify = require('gulp-uglify');  var sass = require('gulp-ruby-sass');  var autoprefixer = require('gulp-autoprefixer');  var minifyCss = require('gulp-minify-css');  gulp.task('scripts', function () {  gulp.src('src/js/\*\*/\*.js')  .pipe(concat('scripts.js'))  .pipe(uglify())  .pipe(gulp.dest('dest'));  });  gulp.task('sass', function () {  gulp.src('src/sass/styles.scss')  .pipe(sass())  .pipe(autoprefixer())  .pipe(minifyCss())  .pipe(gulp.dest('dest'));  });  gulp.task('default', ['scripts', 'sass'], function () {  gulp.watch('src/js/\*\*/\*.js', ['scripts']);  gulp.watch('src/sass/\*\*/\*.{sass,scss}', ['sass']);  }); | | |
| Both tasks involve a task per file. This approach literally involves moving a task to each file and is the approach detailed in the recipe in the Gulp repository: | | |
| gulp-tasks/scripts.js:  var gulp = require('gulp');  var concat = require('gulp-concat');  var uglify = require('gulp-uglify');  gulp.task('scripts', function () {  gulp.src('src/js/\*\*/\*.js')  .pipe(concat('scripts.js'))  .pipe(uglify())  .pipe(gulp.dest('dest'));  }); | | gulp-tasks/sass.js:  var gulp = require('gulp');  var sass = require('gulp-ruby-sass');  var autoprefixer = require('gulp-autoprefixer');  var minifyCss = require('gulp-minify-css');  gulp.task('sass', function () {  gulp.src('src/sass/styles.scss')  .pipe(sass())  .pipe(autoprefixer())  .pipe(minifyCss())  .pipe(gulp.dest('dest'));  }); |
| Then you can use the require-dir module to include every file in the gulp-tasks directory into your gulpfile.  The following is all you would need in your gulpfile:  var gulp = require('gulp');  var requireDir = require('require-dir');  requireDir('./gulp-tasks');  gulp.task('default', ['scripts', 'sass'], function () {  gulp.watch('src/js/\*\*/\*.js', ['scripts']);  gulp.watch('src/sass/\*\*/\*.{sass,scss}', ['sass']);  });  You can move the default task out into its own file, but I often prefer not to.  You can shorten the two require-dir lines to a single line, too:  require('require-dir')('./gulp-tasks');  This approach is good on simple gulpfiles, but if you want to do more complicated things like store your source and destination file paths in an object or use gulp-load-plugins, then this approach would lead to a lot of redundencies. | | |
| [gulp-tap](https://www.npmjs.com/package/gulp-tap) | Easily tap into a pipeline.  Uses  Some filters like gulp-coffee process all files. What if you want to process all JS and Coffee files in a single pipeline. Use tap to filter out .coffee files and process them through the coffee filter and let JavaScript files pass through.  gulp.src("src/\*\*/\*.{coffee,js}")      .pipe(tap(function(file, t) {          if (path.extname(file.path) === '.coffee') {              return t.through(coffee, []);          }      }))      .pipe(gulp.dest('build'));  What if you want to change content like add a header? No need for a separate filter, just change the content.  tap(function(file) {      file.contents = Buffer.concat([          new Buffer('HEADER'),          file.contents      ]);  });  If you do not return a stream, tap forwards your changes. | | |
|  | file:[ 'history', 'cwd', 'base', 'stat', '\_contents' ]  file.stat.size | | |
| [run-sequence](https://www.npmjs.com/package/run-sequence) | var gulp = require('gulp');  var runSequence = require('run-sequence');  var del = require('del');  var fs = require('fs');    // This will run in this order:  // \* build-clean  // \* build-scripts and build-styles in parallel  // \* build-html  // \* Finally call the callback function  gulp.task('build', function(callback) {    runSequence('build-clean',                ['build-scripts', 'build-styles'],                'build-html',                callback);  }); | | |
|  | **使用gulp-plumber来捕获处理任务中的错误** 在gulp的管道流任务处理中，如果某个环节出了错，会导致整个任务中断，包括watch任务，这很麻烦。所以gulp-plumber来了。   1. var plumber = require('gulp-plumber'); 2. var less= require('gulp-less'); 3. gulp.src('./src/\*.ext') 4. .pipe(plumber()) 5. .pipe(less()) 6. .pipe(gulp.dest('./dist')); | | |
| gulp-changed | 使用gulp-changed来过滤变动的文件  默认情况下，每次运行时候所有的文件都会传递并通过整个管道。通过使用 [gulp-changed](https://github.com/sindresorhus/gulp-changed" \t "http://www.gulpjs.com.cn/docs/recipes/only-pass-through-changed-files/_blank) 可以只让更改过的文件传递过管道。这可以大大加快连续多次的运行。  // npm install --save-dev gulp gulp-changed gulp-jscs gulp-uglify  var gulp = require('gulp');  var changed = require('gulp-changed');  var jscs = require('gulp-jscs');  var uglify = require('gulp-uglify');  // 我们在这里定义一些常量以供使用  var SRC = 'src/\*.js';  var DEST = 'dist';  gulp.task('default', function() {  return gulp.src(SRC)  // `changed` 任务需要提前知道目标目录位置  // 才能找出哪些文件是被修改过的  .pipe(changed(DEST))  // 只有被更改过的文件才会通过这里  .pipe(jscs())  .pipe(uglify())  .pipe(gulp.dest(DEST));  });  changed的参数中输出文件夹这一参数不能省，而且不能和源文件夹相同。[gulp-watch](https://www.npmjs.com/package/gulp-watch" \t "http://www.qiqiboy.com/post/_blank) 也可以实现类似效果，而且不需要指定输出文件夹，但是首次使用文件变动它无法正确处理到。 | | |
| del | var gulp = require('gulp');  var del = require('del'); // rm -rf  gulp.task('clean', function (cb) {  del(['output'], cb);  });  gulp.task('templates', ['clean'], function () {  return gulp.src(['src/templates/\*.hbs'])  .pipe(gulp.dest('output/templates/'));  // 返回一个 stream 来表示它已经被完成  });  gulp.task('styles', ['clean'], function () {  return gulp.src(['src/styles/app.less'])  .pipe(gulp.dest('output/css/app.css'));  });  gulp.task('build', ['templates', 'styles']);  // templates 和 styles 将会并行处理  // clean 将会保证在任一个任务开始之前完成  // clean 并不会被执行两次，尽管它被作为依赖调用了两次  gulp.task('default', ['build']); | | |
|  |  | | |
| 自动加载插件[gulp-load-plugins](https://www.npmjs.com/package/gulp-load-plugins) | 使用[gulp-load-plugins](https://www.npmjs.com/package/gulp-load-plugins)  安装：npm install --save-dev gulp-load-plugins  要使用gulp的插件，首先得用require来把插件加载进来，如果我们要使用的插件非常多，那我们的gulpfile.js文件开头可能就会是这个样子的：  var gulp = require('gulp'),  //一些gulp插件,abcd这些命名只是用来举个例子  a = require('gulp-a'),  b = require('gulp-b'),  c = require('gulp-c'),  d = require('gulp-d'),  e = require('gulp-e'),  f = require('gulp-f'),  g = require('gulp-g'),  //更多的插件...  z = require('gulp-z');  虽然这没什么问题，但会使我们的gulpfile.js文件变得很冗长，看上去不那么舒服。gulp-load-plugins插件正是用来解决这个问题。  gulp-load-plugins这个插件能自动帮你加载package.json文件里的gulp插件。例如假设你的package.json文件里的依赖是这样的:  {  "devDependencies": {  "gulp": "~3.6.0",  "gulp-rename": "~1.2.0",  "gulp-ruby-sass": "~0.4.3",  "gulp-load-plugins": "~0.5.1"  }  }  然后我们可以在gulpfile.js中使用gulp-load-plugins来帮我们加载插件：  var gulp = require('gulp');//加载gulp-load-plugins插件，并马上运行它  var plugins = require('gulp-load-plugins')();  然后我们要使用gulp-rename和gulp-ruby-sass这两个插件的时候，就可以使用plugins.rename和plugins.rubySass来代替了,也就是原始插件名去掉gulp-前缀，之后再转换为驼峰命名。  var gulp = require('gulp'),  gulpLoadPlugins = require('gulp-load-plugins'),  plugins = gulpLoadPlugins();  gulp.task('js', function () {  return gulp.src('js/\*.js')  .pipe(plugins.jshint())  .pipe(plugins.jshint.reporter('default'))  .pipe(plugins.uglify())  .pipe(plugins.concat('app.js'))  .pipe(gulp.dest('build'));  });  实质上gulp-load-plugins是为我们做了如下的转换  plugins.rename = require('gulp-rename');  plugins.rubySass = require('gulp-ruby-sass');  gulp-load-plugins并不会一开始就加载所有package.json里的gulp插件，而是在我们需要用到某个插件的时候，才去加载那个插件。  最后要提醒的一点是，因为gulp-load-plugins是通过你的package.json文件来加载插件的，所以必须要保证你需要自动加载的插件已经写入到了package.json文件里，并且这些插件都是已经安装好了的。 | | |
|  |  | | |
|  |  | | |

YUI Compressor

|  |  |
| --- | --- |
|  | http://yui.github.io/yuicompressor/ |
|  | java -jar yuicompressor-x.y.z.jar  Usage: java -jar yuicompressor-x.y.z.jar [options] [input file]  Global Options  -h, --help Displays this information  --type <js|css> Specifies the type of the input file  --charset <charset> Read the input file using <charset>  --line-break <column> Insert a line break after the specified column number  -v, --verbose Display informational messages and warnings  -o <file> Place the output into <file> or a file pattern.  Defaults to stdout.  JavaScript Options  --nomunge Minify only, do not obfuscate  --preserve-semi Preserve all semicolons  --disable-optimizations Disable all micro optimizations  GLOBAL OPTIONS  -h, --help  Prints help on how to use the YUI Compressor  --line-break  Some source control tools don't like files containing lines longer than,  say 8000 characters. The linebreak option is used in that case to split  long lines after a specific column. It can also be used to make the code  more readable, easier to debug (especially with the MS Script Debugger)  Specify 0 to get a line break after each semi-colon in JavaScript, and  after each rule in CSS.  --type js|css  The type of compressor (JavaScript or CSS) is chosen based on the  extension of the input file name (.js or .css) This option is required  if no input file has been specified. Otherwise, this option is only  required if the input file extension is neither 'js' nor 'css'.  --charset character-set  If a supported character set is specified, the YUI Compressor will use it  to read the input file. Otherwise, it will assume that the platform's  default character set is being used. The output file is encoded using  the same character set.  -o outfile  Place output in file outfile. If not specified, the YUI Compressor will  default to the standard output, which you can redirect to a file.  Supports a filter syntax for expressing the output pattern when there are  multiple input files. ex:  java -jar yuicompressor.jar -o '.css$:-min.css' \*.css  ... will minify all .css files and save them as -min.css  -v, --verbose  Display informational messages and warnings.  JAVASCRIPT ONLY OPTIONS  --nomunge  Minify only. Do not obfuscate local symbols.  --preserve-semi  Preserve unnecessary semicolons (such as right before a '}') This option  is useful when compressed code has to be run through JSLint (which is the  case of YUI for example)  --disable-optimizations  Disable all the built-in micro optimizations.  Note: If no input file is specified, it defaults to stdin.  The following command line (x.y.z represents the version number):  $ java -jar yuicompressor-x.y.z.jar myfile.js -o myfile-min.js  will minify the file myfile.js and output the file myfile-min.js. For more information on how to use the YUI Compressor, please refer to the documentation included in the archive.  The charset parameter isn't always required, but the compressor may throw an error if the file's encoding is incompatible with the system's default encoding. In particular, if your file is encoded in utf-8, you should supply the parameter.  $ java -jar yuicompressor-x.y.z.jar myfile.js -o myfile-min.js --charset utf-8 |

使用案例

|  |  |
| --- | --- |
|  | gulp自动化构建工具学习  <http://www.jianshu.com/p/00b562362fec> |
|  | [Gulp的配置文件gulpfile.babel.js](http://blog.csdn.net/itpinpai/article/details/52872129)  <http://blog.csdn.net/SeekerTime/article/details/70325162>  <http://blog.csdn.net/itpinpai/article/details/52872129> |
|  | [gulpfile.js 中的坑--注入篇（gulp-inject,gulp-wiredep）](http://blog.csdn.net/faithfairy/article/details/50249479)  http://blog.csdn.net/faithfairy/article/details/50249479 |
|  | 一份gulp的配置文件  <http://blog.csdn.net/qq_15096707/article/details/53582319>  <http://www.cnblogs.com/QRL909109/p/5620824.html>  [Gulp编译、合并、压缩，以及Browsersync实时刷新教程](http://blog.csdn.net/beverley__/article/details/55213235)  <http://blog.csdn.net/beverley__/article/details/55213235>  **[Gulp的配置文件gulpfile.babel.js](http://blog.csdn.net/itpinpai/article/details/52872129)**  <http://blog.csdn.net/itpinpai/article/details/52872129?locationNum=3&fps=1> |
|  |  |
|  |  |
|  |  |