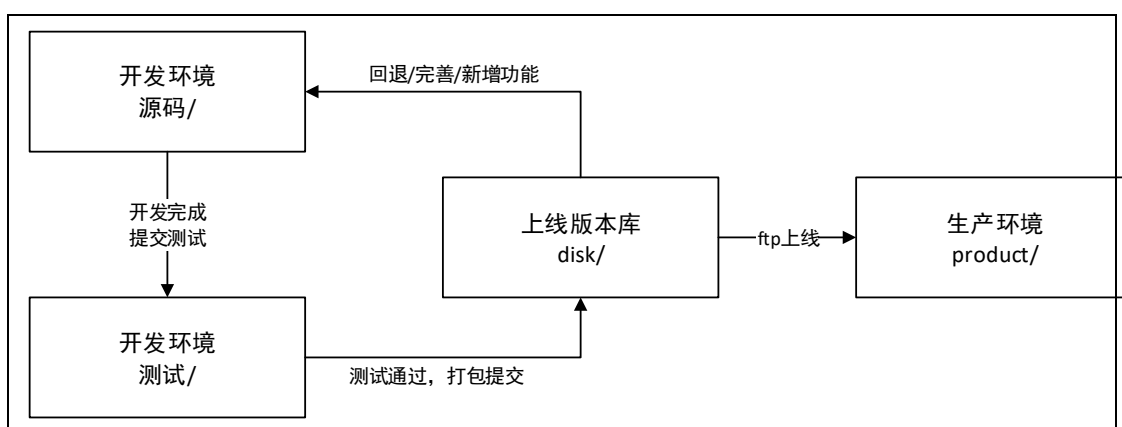


# Gulp 教程

## 1-1 简介

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

它不仅能对网站资源进行优化，而且在开发过程中很多重复任务能够使用正确的工具自动完成，使用它，不仅可以很愉快的编写代码，而且能大大提升开发效率



正式项目生产线上线流程，开发环境中进行源码开发，开发完成后提交测试。

测试通过的项目经过**打包提交**，提交合适的版本到版本库中，由部署人员提交到生产线上进行运行。

最后打包提交的一部分，要进行源码的优化和压缩，使用 gulp 能极大的自动化的进行处理和优化。

gulp 是基于 nodeJS 的自动任务运行器，它能自动化的完成 javascript/coffee/sass/less/html/images/css 等文件的测试、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。  
gulp 和 grunt 都是自动化的工具，grunt 采取的频繁 IO 操作，相比 gulp 的流操作，gulp 能更快的更便捷的完成构建工作。



## 1-2 下载安装

下载安装使用 gulp

首先安装 nodejs, 通过 nodejs 的 npm 全局安装和项目安装 **gulp 全局插件**, 其次在项目里安装所需要的 **gulp 插件**, 然后新建 gulp 的配置文件 **gulpfile.js** 配置文件定制 **gulp 任务**, 通过命令行运行 gulp 任务即可。

安装 nodeJS

- ➔ 全局安装 gulp
- ➔ 项目中安装 gulp 以及 gulp 插件
- ➔ 配置 gulpfile.js
- ➔ 运行任务

安装 nodeJS: <https://nodejs.org/en/>

下载安装包, 安装。

命令行运行 node -v 测试安装

```
C:\Windows\system32>node -v  
v6.8.0
```

安装 gulp 插件

```
>> npm install gulp -g // 全局安装 gulp 插件
```

备注：npm 命令

npm: node package manager, nodejs 包管理器, 用于 node 插件管理 (安装、卸载、管理依赖)

npm 安装插件: `npm install <name> [-g] [--save-dev]`

- ➔ **<name>**: 插件的名称
- ➔ **-g**: 全局安装, 将会安装在 `C:\Users\Administrator\AppData\Roaming\npm`, 并且写入系统环境变量。非全局安装, 就会安装在当前目录下 `node_module/` 下, 通过 **require** 引入
- ➔ **--save**: 将配置信息保存到 `package.json` 文件中【`package.json` 是 Node 项目配置文件】
- ➔ **-dev**: 保存配置到 `devDependencies` 节点, 不指定就会保存 `dependencies` 节点

node 插件比较大, 不方便加入版本管理中, 于是将依赖插件的信息保存在 `package.json` 文件中, 这个文件可以交给版本管理, 其他人只需要获取 `package.json`, 就可以在自己本地通过 `npm install` 命令根据 `package.json` 中的依赖信息, 自行下载引入插件。

卸载插件: `npm uninstall <name> [-g] [--save-dev]`

删除全部插件: `npm install rimraf -g | rimraf node_modules`

更新插件: `npm update <name> [--save-dev]`

更新全部插件: `npm update [--save-dev]`

查看 npm 帮助: `npm help`

查看已安装列表: `npm list`



## 1-3 Quick start

需求：完成一个自动对项目 js 打包的任务配置，整合文件到一个文件中。

### ✓ 第一步：创建项目文件夹[gulp\_pro/]，完善项目结构



The screenshot shows a Windows File Explorer window with the address bar set to 'Software (D:) > WORK\_RESOURCE > gulp\_pro'. The file list includes folders for 'css', 'disk', 'images', and 'js', along with an 'index.html' file. Red boxes highlight the 'disk' folder and the 'css', 'images', and 'js' folders. Red arrows point from these boxes to text labels: '最终版本存放目录' (Final version storage directory) for 'disk' and 'gulp编译目标目录' (Gulp compilation target directory) for the other folders.

名称	修改日期	类型	大小
css	2016/10/...	文件夹	
disk	2016/10/...	文件夹	
images	2016/10/...	文件夹	
js	2016/10/...	文件夹	
index.html	2016/10/...	文件	0 KB

css/: 样式开发目录  
js/: 脚本开发目录  
images/: 开发中图片目录  
disk/: 编译后的文件存放目录，gulp 运行后的所有文件会存放在这里

### ✓ 第二步：进入项目文件夹，初始化项目，自动生成 package.json 文件

```
D:\WORK_RESOURCE\gulp_pro>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (gulp_pro) gulp_pro
version: (1.0.0) 0.0.1
description: 这是一个测试项目
entry point: (index.js)
test command:
git repository:
keywords:
author: 牟文斌
license: (ISC) ISC
About to write to D:\WORK_RESOURCE\gulp_pro\package.json:
{
  "name": "gulp_pro",
  "version": "0.0.1",
  "description": "这是一个测试项目",
  "main": "index.js",
  "dependencies": {
    "gulp": "^3.9.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo `Error: no test specified` && exit 1"
  },
  "author": "牟文斌",
  "license": "ISC"
}

Is this ok? (yes)
D:\WORK_RESOURCE\gulp_pro>
```



打开命令行窗口，进入目标文件夹，键入 `npm init` 初始化项目

`name:(gulp_pro)` 输入项目名称（不输入就使用当前目录名称）

`version:(1.0.0)` 输入版本号，默认是括号中的版本号

`description:` 项目描述信息

`entry point:(index.js)` 目标文件

`test command:` 测试命令，可以直接回车表示忽略

`git repository:` git 仓库，可以忽略

`keyword:` 关键词

`author:` 作者

`license:` 授权信息

### ✓ 第三步：安装 gulp 插件，安装文件合并 gulp-concat 插件

安装 gulp 插件

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp --save-dev
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Ple
0 as soon as possible. Use 'npm ls graceful-fs' to find it in the tree.
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp@3.9.1
npm WARN gulp_pro@0.0.1 No repository field.
```

安装 gulp-concat 插件

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-concat --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-concat@2.6.0
+-- concat-with-sourcemaps@1.0.4
|   -- source-map@0.5.6
|   -- through2@0.6.5
|   -- readable-stream@1.0.34
npm WARN gulp_pro@0.0.1 No repository field.
```

执行命令：`npm install gulp --save-dev`

未完成后再 `package.json` 中，就会出现 `devDependencies` 配置项

执行命令：`npm install gulp-concat --save-dev`

gulp-concat 合并文件插件

```
9   "devDependencies": {
10     "gulp": "^3.9.1",
11     "gulp-concat": "^2.6.0"
12   },
```

### ✓ 第四步：创建 gulpfile.js 配置文件，配置信息

```
1 // 导入工具包
2 var gulp = require('gulp');
3 var concat = require('gulp-concat');
4
5 gulp.task('build', function() { // 创建任务，任务名称build
6   gulp.src(['js/index.js', 'js/main.js']) // 指定要编译的文件
7     .pipe(concat('product.js')) // 执行压缩，和前面引入的工具包名称一致
8     .pipe(gulp.dest('disk/js/')); // 任务执行后的文件
9 });
```



.src():要打包的所有文件列表

.pipe():管道符, 使用.pipe()前面的命令作为参数执行 pipe()后面指定的命令

.dest():打包提交的目标文件夹

正在 gulp/js/目录下创建 index.js 和 main.js 文件

名称	修改日期	类型	大小
index.js	2016/10/...	JetBrains ...	1 KB
main.js	2016/10/...	JetBrains ...	1 KB

index.js

```
1 $(function() {  
2     console.log("index.js文件中的脚本");  
3 })
```

main.js

```
1 $(function() {  
2     console.log("mian.js文件中的脚本");  
3 })
```

#### ✓ 第五步：命令行运行 gulp 任务

```
D:\WORK_RESOURCE\gulp_pro>gulp build  
[22:46:51] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js  
[22:46:51] Starting 'build'...  
[22:46:51] Finished 'build' after 14 ms  
D:\WORK_RESOURCE\gulp_pro>
```

自动编译生成的文件

Software (D:) > WORK_RESOURCE > gulp_pro > disk > js			
名称	修改日期	类型	大小
product.js	2016/10/...	JetBrains ...	1 KB

生成文件的内容

```
1 $(function() {  
2     console.log("index.js文件中的脚本");  
3 })  
4  
5  
6 $(function() {  
7     console.log("index.js文件中的脚本");  
8 })
```

## 1-4 Gulp 插件

Gulp 本身是一个工作环境, 提供了基本的处理模块, 常规的其他操作, 都是依

官方网的: <http://gulpjs.com>

中文网站: <http://www.gulpjs.com.cn>



赖插件来进行完成的。本节主要从两方面入手进行分析：

第一方面，了解 Gulp 的基本操作 API

第二方面，了解并掌握 gulp 的常见插件的操作

## 1) 常规 API

### `gulp.src(globs [,options])`

描述：src 方法是指定需要处理的源文件的路径。

参数：

globs:需要处理的源文件的路径，必填

src/a.js：指定具体文件

src/\*.js：匹配所有文件，这里匹配 src 下所有的 js 文件

src/\*\*/：匹配 0 个或者多个子文件夹，这里匹配 src 下所有文件夹

src{a,b}.js：匹配多个属性，这里匹配 src 下 a.js 和 b.js

!src/a.js：排除文件，这里匹配结果中排除掉 a.js 文件

options:类型，可选，有 3 个属性：buffer、read、base

options.buffer:boolean 类型，默认 true

options.read:boolean 类型，默认 true

options.base:string 类型，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

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

描述：dest 方法指定处理完成后文件的输出路径

参数：

path:指定文件输出路径，或者定义函数返回输出路径

options:有两个属性 cwd、mode

options.cwd:string 类型，默认 process.cwd()前脚本的工作目录路径

options.mode:string 类型，默认 0777，指定被创建的文件夹的权限

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

描述：task 定义一个可执行的 gulp 任务

参数：

name:string 类型，任务名称（不能包含空格）

deps:该任务依赖的任务，是一个 string 类型的数组，执行完数组中任务之后才会执行当前任务

fn:任务调用执行完成后的回调函数

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

描述：watch 方法用于监听文件变化，一旦变化就会执行指定任务

参数：

glob:需要处理的源文件匹配的路径，可以使一个字符串表示的路径也可以使数组【必填】

tasks:需要执行的任务的名称数组【必填】

cb:每个文件变化执行的回调函数【可选】



## 2) 常见插件

### ● Sass 编译插件——gulp-sass

1) 安装插件：npm install gulp-sass --save-dev

```
gulp pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-sass@2.3.2
+-- node-sass@3.10.1
|   +-- async-foreach@0.1.3
|   +-- cross-spawn@3.0.1
|   |   -- lru-cache@4.0.1
|   |   +-- pseudomap@1.0.2
|   |   -- yallist@2.0.0
|   +-- gaze@1.1.2
|   |   -- globule@1.0.0
|   |   +-- glob@7.0.6
|   |   -- lodash@4.9.0
```

2) 编辑 src/sass/index.scss

```
index.scss
1 @charset "utf-8";
2
3 $base-font-size:14px;
4 $base-color:#333;
5
6 body{
7     font-size:$base-font-size;
8     color:$base-color;
9 }
```

3) 配置 gulpfile.js 文件，添加 sass 任务

```
11 var sass = require('gulp-sass');
12
13 gulp.task('sass', function() {
14     return gulp.src('src/sass/*.scss') // 指定源文件目录
15         .pipe(sass({style:'expanded'})) // 使用sass()插件进行编译压缩
16         .pipe(gulp.dest('css')); // 生成到css目录下
17 })
```

4) 执行任务:gulp sass

```
D:\WORK_RESOURCE\gulp_pro>gulp sass
[12:05:13] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[12:05:13] Starting 'sass'...
[12:05:13] Finished 'sass' after 10 ms
```

5) 执行完成

```
index.css
1 body {
2     font-size: 14px;
3     color: #333; }
```



## ● Less 编译插件——gulp-less

1) 安装插件: `npm install gulp-less --save-dev`

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-less --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-less@3.1.0
+-- accord@0.23.0
|   +-- convert-source-map@1.3.0
|   +-- fobject@0.0.4
|   |   -- semver@5.3.0
+-- glob@7.0.3
```

2) 编辑 `src/js/less/demo.less`

```
demo.less
1 @charset "utf-8";
2
3 @base-font-size:14px;
4 @base-color:#fff;
5
6 body{
7     font-size:@base-font-size;
8     color:@base-color;
9 }
```

3) 配置 `gulpfile.js` 文件, 添加 less 任务

```
19 var less = require("gulp-less");
20
21 gulp.task("less", function() { // 创建less任务
22     return gulp.src('src/less/*.less') // 指定源文件目录
23         .pipe(less()) // 执行插件进行编译
24         .pipe(gulp.dest("disk/js")); // 指定目标文件目录
25 });
```

4) 执行任务: `gulp less`

```
D:\WORK_RESOURCE\gulp_pro>gulp less
[14:05:51] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:05:51] Starting 'less'...
[14:05:51] Finished 'less' after 9.25 ms
```

5) 查看结果

```
demo.less  demo.css
1 @charset "utf-8";
2 body {
3     font-size: 14px;
4     color: #fff;
5 }
```





## ● 文件合并插件——gulp-concat

1) 安装插件: `npm install gulp-concat --save-dev`

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-concat --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-concat@2.6.0
```

2) 编辑 `src/js/demo.js` & `index.js`

参考 quick start

3) `gulpfile.js` 中配置任务

参考 quick start

4) 执行任务: `gulp concat`

参考 quick start

5) 结果查看

参考 quick start

## ● JS 压缩插件——gulp-uglify

1) 安装插件

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-uglify --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-uglify@2.0.0
  +--- lodash@4.16.4
  +--- make-error-cause@1.2.1
  |   +--- make-error@1.2.1
  |   +--- uglify-js@2.7.0
  |       +--- yargs@3.10.0
  |           +--- camelcase@1.2.1
  |           +--- cliui@2.1.0
  |           +--- window-size@0.1.0
  |           +--- uglify-save-license@0.4.1
```

2) 编辑 `src/js/main.js`

```
main.js
1 $(function() {
2     $(".container").on("click", ".btn", function() {
3         console.log("按键被点击了...");
4     })
5 })
```

3) `gulpfile.js` 中配置任务

```
27 var uglify = require("gulp-uglify");// 引入模块
28
29 gulp.task("uglify", function() { // 定义任务
30     return gulp.src("src/js/*.js") // 指定源文件
31         .pipe(uglify()) // 执行压缩
32         .pipe(gulp.dest("disk/js")); // 输出目录
33 })
```

4) 执行任务: `gulp uglify`

```
D:\WORK_RESOURCE\gulp_pro>gulp uglify
[14:16:41] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:16:41] Starting 'uglify'...
[14:16:41] Finished 'uglify' after 9.45 ms
```

5) 结果查看

```
main.js - src/js  main.js - disk/js
1 $(function(){ $(".container").on("click", ".btn", function(){ console.log("按键被点击了...") }); });
```



- 重命名插件——gulp-rename

1) 安装插件: `npm install gulp-rename --save-dev`

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-rename --sve-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-rename@1.2.2
```

1) 安装插件: npm insta

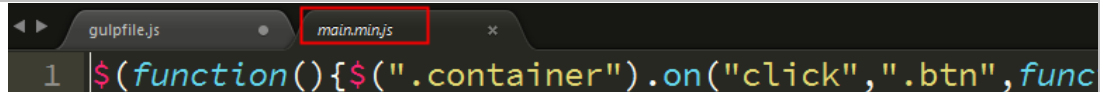
2) gulpfile.js 中配置任务

```
35 var rename = require("gulp-rename");
36
37 gulp.task("uglify-min", function() {
38     return gulp.src("src/js/main.js")// 指定源码文件
39         .pipe(rename("main.min.js"))// 重命名文件
40         .pipe(gulp.dest("disk/js")); // 保存文件
41 })
```

3) 执行任务

```
D:\WORK_RESOURCE\gulp_pro>gulp uglify-min
[14:28:29] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:28:29] Starting 'uglify-min'...
[14:28:29] Finished 'uglify-min' after 11 ms
```

4) 结果查看



- 样式压缩插件——gulp-minify-css

#### 1) 安装插件: npm install gulp-minify-css

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-minify-css
npm WARN deprecated gulp-minify-css@1.2.4: Please use gulp-clean-css
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-minify-css@1.2.4
+-- clean-css@3.4.20
+-- commander@2.8.1
+-- source-map@0.4.4
+-- amdefine@1.0.0
+-- object-assign@4.1.0
+-- readable-stream@2.1.5
+-- buffer-shims@1.0.0
+-- isarray@1.0.0
+-- vinyl-bufferstream@1.0.1
+-- bufferstreams@1.0.1
```

#### 2) 编辑 src/css/main.css

```
1 *{
2   margin:0;
3   padding:0;
4 }
5 body{
6   font-size:14px;
7   color:#333;
8 }
```

#### 3) gulpfile.js 中配置任务

```
43 var minifyCss = require("gulp-minify-css");// 引入插件模块
44
45 gulp.task("minifycss", function() { // 创建任务
46   return gulp.src("src/css/*.css") // 指定源文件目录
47     .pipe(minifyCss()) // 执行压缩
48     .pipe(gulp.dest("disk/css")) // 指定输出目标文件夹
49 })
```

#### 4) 执行任务 : gulp minifycss

```
D:\WORK_RESOURCE\gulp_pro>gulp minifycss
[14:37:40] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:37:40] Starting 'minifycss'...
[14:37:40] Finished 'minifycss' after 14 ms
```

#### 5) 结果查看

```
1 *{margin:0;padding:0}body{font-size:14px;color:#333}
```



## ● 图片压缩插件——gulp-imagemin

### 1) 安装插件: npm install gulp-imagemin

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-imagemin --save-dev
npm WARN deprecated cross-spawn-async@2.2.4: cross-spawn no longer requires a build toolchain
> gifsicle@3.0.4 postinstall D:\WORK_RESOURCE\gulp_pro\node_modules\gifsicle
> node lib/install.js
< gifsicle was build test passed successfully
```

### 2) 在 images/ 目录下存放图片

名称	修改日期	类型	大小
qrcode for gh c5c0b3b7fa88 258.jpg	2016/10/...	JPG 文件	28 KB
qrcode for gh c5c0b3b7fa88 344.jpg	2016/10/...	JPG 文件	9 KB
qrcode for gh c5c0b3b7fa88 430.jpg	2016/10/...	JPG 文件	40 KB
qrcode for gh c5c0b3b7fa88 860.jpg	2016/10/...	JPG 文件	100 KB
qrcode for gh c5c0b3b7fa88 1280.jpg	2016/10/...	JPG 文件	149 KB

### 3) gulpfile.js 中配置任务

```
51 var imagemin = require("gulp-imagemin");// 引入模块
52
53 gulp.task("imagemin", function() { // 创建任务
54     return gulp.src("images/**/*.jpg") // 指定源文件目录
55         .pipe(imagemin()) // 执行压缩
56         .pipe(gulp.dest("disk/images")) // 指定输出目录
57 });
```

### 4) 执行任务: gulp imagedin

```
D:\WORK_RESOURCE\gulp_pro>gulp imagedin
[14:43:49] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:43:49] Starting 'imagemin'...
[14:43:49] Finished 'imagemin' after 9.9 ms
[14:43:49] gulp-imagemin: Minified 5 images (saved 69.19 kB - 20.9%)
```

压缩比例

### 5) 结果查看

#### 压缩前



5 个文件, 0 个文件夹

类型: 类型均为 JPG 文件

位置: 全部位于 D:\WORK\_RESOURCE

大小: 323 KB (331,170 字节)

占用空间: 332 KB (339,968 字节)

#### 压缩后



5 个文件, 0 个文件夹

类型: 类型均为 JPG 文件

位置: 全部位于 D:\WORK\_RESOURCE\gulp\_pro\di

大小: 255 KB (261,985 字节)

占用空间: 264 KB (270,336 字节)



- 服务器插件——gulp-connect

1) 安装插件：npm install gulp-connect --save-dev

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-connect --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-connect@5.0.0
  +- connect@2.30.2
  +- basic-auth-connect@1.0.0
  +- body-parser@1.13.3
  +- iconv-lite@0.4.11
  +- on-finished@2.3.0
  +- ee-first@1.1.1
```

3) gulpfile.js 中配置任务

4) 执行任务

2) gulpfile.js 中配置任务

```
59 var connect = require("gulp-connect");// 引入模块
60
61 gulp.task("server", function() { // 创建任务
62     return connect.server(); // 启动服务
63 });
```

3) 执行任务：gulp server

```
D:\WORK_RESOURCE\gulp_pro>gulp server
[14:56:39] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[14:56:39] Starting 'server'...
[14:56:39] Finished 'server' after 18 ms
[14:56:39] Server started http://localhost:8080
```

4) 结果查看：打开浏览器，访问 http://localhost:8080

localhost:8080

hello gulp!



## ● 浏览器自动刷新——gulp-livereload

1) 安装插件: `npm install gulp-livereload --save-dev`

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-livereload --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-livereload@3.8.1
+-- chalk@0.5.1
+-- ansi-styles@1.1.0
+-- has-ansi@0.1.0
+-- ansi-regex@0.2.1
+-- strip-ansi@0.3.0
+-- supports-color@0.2.0
```

● 浏览器自动刷新——gulp-livereload。

2) 安装插件: `npm install gulp-watch --save-dev`

```
D:\WORK_RESOURCE\gulp_pro>npm install gulp-watch --save-dev
gulp_pro@0.0.1 D:\WORK_RESOURCE\gulp_pro
-- gulp-watch@4.3.10
+-- anymatch@1.3.0
+-- chokidar@1.6.1
+-- async-each@1.0.1
+-- is-binary-path@1.0.1
+-- binary-extensions@1.7.0
+-- readdirp@2.1.0
```

4) 执行任务。

5) 结果查看。

3) gulpfile.js 中配置任务

```
1 var gulp = require("gulp"), // 引入主要模块
2   connect = require("gulp-connect"), // 引入服务模块
3   reload = require("gulp-livereload"), // 引入重加载模块
4   watch = require("gulp-watch"); // 引入监听模块
5
6 gulp.task("copy-index", function() { // 创建任务—拷贝首页
7   return gulp.src('index.html')
8     .pipe(gulp.dest("disk/"))
9     .pipe(connect.reload());
10 });
11
12 gulp.task("sass", function() { // 创建任务，自动编译sass文件
13   return gulp.src('src/sass/**/*.scss')
14     .pipe(sass())
15     .pipe(gulp.dest("disk/js/"));
16 });
17
18 gulp.task("watch", function() { // 创建任务，自动监听
19   gulp.watch("index.html", ['copy-index']);
20   gulp.watch("src/sass/**/*.scss", ["sass"]);
21 })
22
23 gulp.task("server", function() { // 创建任务，启动服务
24   return connect.server({
25     root:"disk",
26     livereload:true
27   });
28 })
29
30 gulp.task("default", ['server', 'watch']);
```



#### 4) 执行任务

```
D:\WORK_RESOURCE\gulp_pro>gulp
[15:40:02] Using gulpfile D:\WORK_RESOURCE\gulp_pro\gulpfile.js
[15:40:02] Starting 'server'...
[15:40:02] Finished 'server' after 15 ms
[15:40:02] Starting 'watch'...
[15:40:02] Finished 'watch' after 15 ms
[15:40:02] Starting 'default'...
[15:40:02] Finished 'default' after 20 μs
[15:40:02] Server started http://localhost:8080
[15:40:02] LiveReload started on port 35729
[15:41:03] Starting 'copy-index'...
[15:41:04] Finished 'copy-index' after 80 ms
[15:41:12] Starting 'copy-index'...
[15:41:12] Finished 'copy-index' after 12 ms
```

#### 5) 结果查看

修改 index.html 页面中的内容，在保存内容的同时，网页上实时刷新显示新的内容。



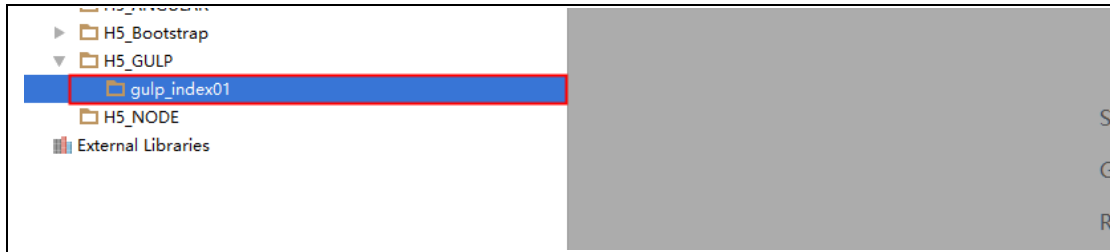
#### ● 备注：

- gulp-watch: 监听模块，根据监听的文件，执行指定的任务列表
- gulp.task("default"..:default 任务，gulp 中的默认任务

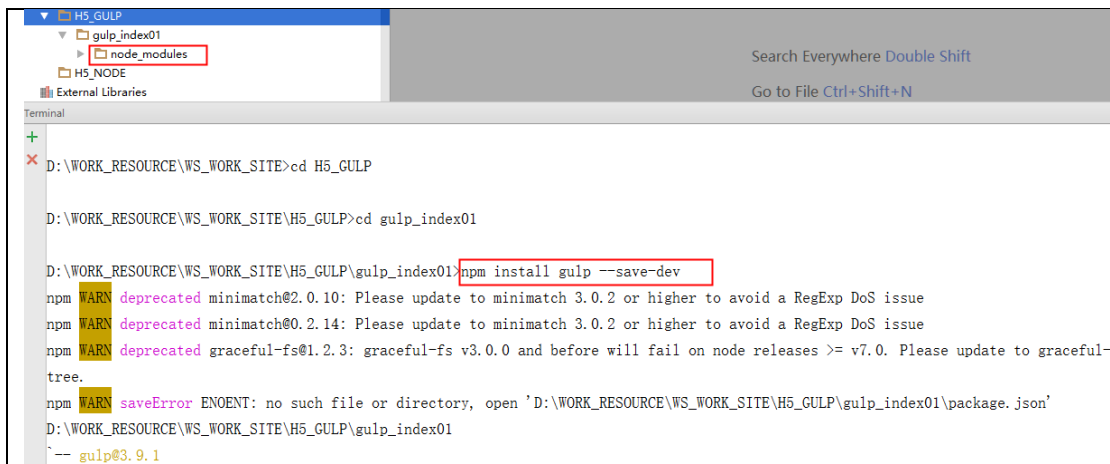


## 1-5 GULP 在 WEBSTORM 中的使用

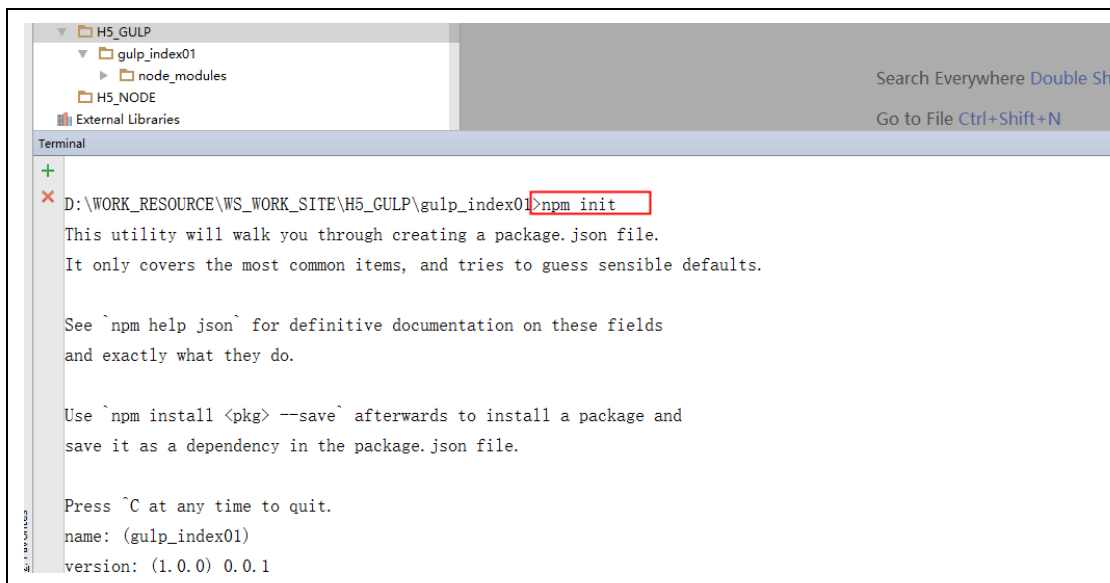
### 1) 创建项目



### 2) 命令行添加 gulp 插件

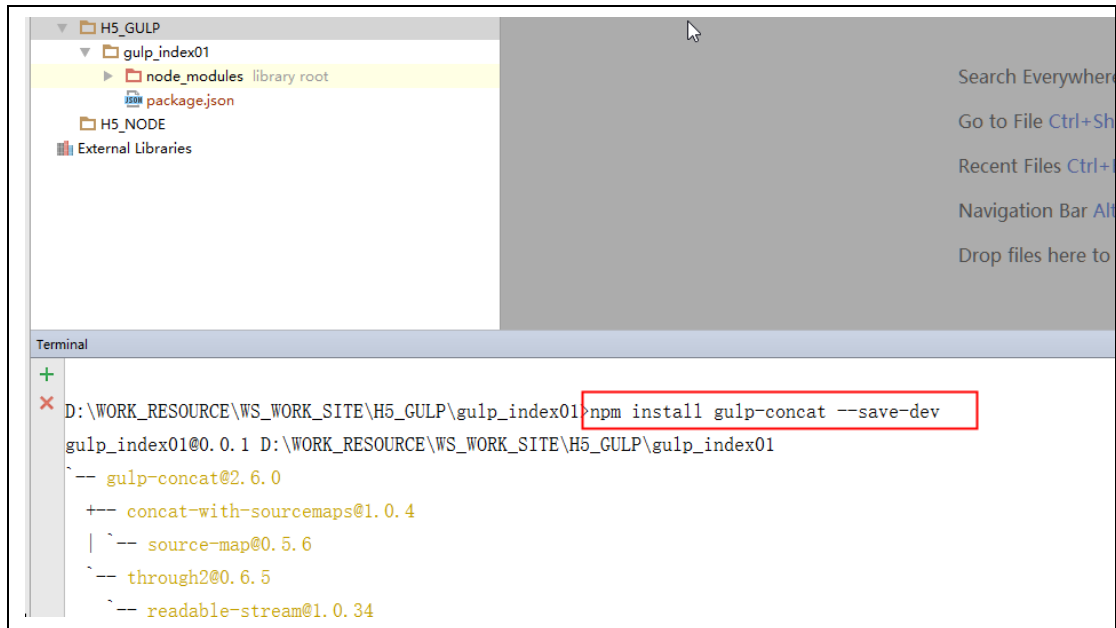


### 3) 创建 package.json 文件【命令行创建】





#### 4) 添加插件——gulp, gulp-concat



#### 5) 创建项目任务文件 gulpfile.js 并定义任务



#### 6) 查看并执行 gulp 任务

- 在 `gulpfile.js` 文件上右键点击
- 弹出的菜单中选择 `show gulp tasks`
- 项目文件中就会出来一个 `gulp task` 窗口，窗口中是 `gulp` 任务列表
- 在任务列表上，右键点击，可以选择执行任务



