1.安装gulp

npm install -g gulp

2.压缩js文件

①安装gulp-uglify模块：npm install gulp-uglify

②新建gulpfile.js文件，其中编写代码：

③获取gulp：var gulp = require('gulp');

④获取gulp-uglify组件：var uglify = require('gulp-uglify');

⑤创建压缩任务：

// 压缩 js 文件

// 在命令行使用 gulp script 启动此任务

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

// 1. 找到文件

gulp.src('js/\*.js')

// 2. 压缩文件

.pipe(uglify())

// 3. 另存压缩后的文件

.pipe(gulp.dest('dist/js'))

})

* gulp.task(name, fn) - 定义任务，第一个参数是任务名，第二个参数是任务内容。
* gulp.src(path) - 选择文件，传入参数是文件路径。
* gulp.dest(path) - 输出文件
* gulp.pipe() - 管道，你可以暂时将 pipe 理解为将操作加入执行队列

⑥终端找到gulpfile.js所在文件，执行gulp script.

⑦检测代码自动执行任务

在 gulpfile.js 中编写如下代码：

// 监听文件修改，当文件被修改则执行 script 任务

gulp.watch('js/\*.js', ['script']);

但是没有命令可以运行 gulp.watch()，需要将 gulp.watch() 包含在一个任务中。

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 script 任务

gulp.watch('js/\*.js', ['script'])

})

接着在命令行输入 gulp auto，自动监听 js/\*.js 文件的修改后压缩js。

⑧使用gulp.task('default',fn)定义默认任务

增加如下代码

gulp.task('default', ['script', 'auto']);

此时你可以在命令行直接输入 gulp +回车，运行 script 和 auto 任务。

⑨最终代码如下：

// 获取 gulp

var gulp = require('gulp')

// 获取 uglify 模块（用于压缩 JS）

var uglify = require('gulp-uglify')

// 压缩 js 文件

// 在命令行使用 gulp script 启动此任务

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

// 1. 找到文件

gulp.src('js/\*.js')

// 2. 压缩文件

.pipe(uglify())

// 3. 另存压缩后的文件

.pipe(gulp.dest('dist/js'))

})

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 script 任务

gulp.watch('js/\*.js', ['script'])

})

// 使用 gulp.task('default') 定义默认任务

// 在命令行使用 gulp 启动 script 任务和 auto 任务

gulp.task('default', ['script', 'auto'])

⑩gulp-watch-path

gulp.watch第二个参数为function时的用法

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

console.log(event);

/\*

当修改 src/js/log.js 文件时

event {

// 发生改变的类型，不管是添加，改变或是删除

type: 'changed',

// 触发事件的文件路径

path: '/Users/nimojs/Documents/code/gulp-book/demo/chapter7/src/js/log.js'

}

\*/

})

我们可以利用 event 给到的信息，检测到某个 js 文件被修改时，只编写当前修改的 js 文件。

可以利用 gulp-watch-path 配合 event 获取编译路径和输出路径。

var watchPath = require('gulp-watch-path')

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

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

var paths = watchPath(event, 'src/', 'dist/')

/\*

paths

{ srcPath: 'src/js/log.js',

srcDir: 'src/js/',

distPath: 'dist/js/log.js',

distDir: 'dist/js/',

srcFilename: 'log.js',

distFilename: 'log.js' }

\*/

gutil.log(gutil.colors.green(event.type) + ' ' + paths.srcPath)

gutil.log('Dist ' + paths.distPath)

gulp.src(paths.srcPath)

.pipe(uglify())

.pipe(gulp.dest(paths.distDir))

})

})

gulp.task('default', ['watchjs'])

**watchPath(event, search, replace, distExt)**

| **参数** | **说明** |
| --- | --- |
| event | gulp.watch 回调函数的 event |
| search | 需要被替换的起始字符串 |
| replace | 第三个参数是新的的字符串 |
| distExt | 扩展名(非必填) |

3.压缩css文件

①安装gulp-minify-css模块

npm install gulp-minify-css

②创建并编写gulpfile.js文件

// 获取 gulp

var gulp = require('gulp')

// 获取 minify-css 模块（用于压缩 CSS）

var minifyCSS = require('gulp-minify-css')

// 压缩 css 文件

// 在命令行使用 gulp css 启动此任务

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

// 1. 找到文件

gulp.src('css/\*.css')

// 2. 压缩文件

.pipe(minifyCSS())

// 3. 另存为压缩文件

.pipe(gulp.dest('dist/css'))

})

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 css 任务

gulp.watch('css/\*.css', ['css'])

});

// 使用 gulp.task('default') 定义默认任务

// 在命令行使用 gulp 启动 css 任务和 auto 任务

gulp.task('default', ['css', 'auto'])

3.压缩图片

①安装gulp-imagemin模块

npm install gulp-imagemin

②创建gulpfile.js文件并编写代码

// 获取 gulp

var gulp = require('gulp');

// 获取 gulp-imagemin 模块

var imagemin = require('gulp-imagemin')

// 压缩图片任务

// 在命令行输入 gulp images 启动此任务

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

// 1. 找到图片

gulp.src('images/\*.\*')

// 2. 压缩图片

.pipe(imagemin({

progressive: true

}))

// 3. 另存图片

.pipe(gulp.dest('dist/images'))

});

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 images 任务

gulp.watch('images/\*.\*)', ['images'])

});

// 使用 gulp.task('default') 定义默认任务

// 在命令行使用 gulp 启动 images 任务和 auto 任务

gulp.task('default', ['images', 'auto'])

4.编译less文件

①安装gulp-less

npm install gulp-less

②创建gulpfile.js文件编写代码

// 获取 gulp

var gulp = require('gulp')

// 获取 gulp-less 模块

var less = require('gulp-less')

// 编译less

// 在命令行输入 gulp less 启动此任务

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

// 1. 找到 less 文件

gulp.src('less/\*\*.less')

// 2. 编译为css

.pipe(less())

// 3. 另存文件

.pipe(gulp.dest('dist/css'))

});

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 less 任务

gulp.watch('less/\*\*.less', ['less'])

})

// 使用 gulp.task('default') 定义默认任务

// 在命令行使用 gulp 启动 less 任务和 auto 任务

gulp.task('default', ['less', 'auto'])

5.编译sass文件

①安装gulp-ruby-sass模块

npm install gulp-ruby-sass

②创建gulpfile.js文件编写代码

// 获取 gulp

var gulp = require('gulp')

// 获取 gulp-ruby-sass 模块

var sass = require('gulp-ruby-sass')

// 编译sass

// 在命令行输入 gulp sass 启动此任务

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

return sass('sass/')

.on('error', function (err) {

console.error('Error!', err.message);

})

.pipe(gulp.dest('dist/css'))

});

// 在命令行使用 gulp auto 启动此任务

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

// 监听文件修改，当文件被修改则执行 images 任务

gulp.watch('sass/\*\*/\*.scss', ['sass'])

});

// 使用 gulp.task('default') 定义默认任务

// 在命令行使用 gulp 启动 sass 任务和 auto 任务

gulp.task('default', ['sass', 'auto'])

6.使用browserify管理依赖

var gulp = require('gulp'),

browserify = require('browserify'),

source = require('vinyl-source-stream');

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

browserify({

//入口点src/main.js

entries : "./src/main.js"

})

//转换为gulp能识别的流

.bundle()

//合并输出为main.js

.pipe(source("main.js"))

//输出到当前目录的build文件夹中

.pipe(gulp.dest("./build/"));

});