link: null
title: 珠峰架构师成长计划
description: gulpfile.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=95 sentences=332, words=2613

1.什么是gulp

- gulp (https://gulpjs.com/)是可以自动化执行任务的工具 在平时开发的流程里面,一定有一些任务需要手工重复得执行，比如:
    - 把文件从开发目录拷贝到生产目录
    - 把多个 JS 或者 CSS 文件合并成一个文件
    - 对JS文件和CSS进行压缩
    - 把sass或者less文件编译成CSS
    - 压缩图像文件
    - 创建一个可以实时刷新页面内容的本地服务器

> 只要你觉得有些动作是要重复去做的,就可以把这些动作创建成一个gulp任务,然后在指定的条件下自动执行

2. gulp特点

- 易于使用 通过代码优于配置的策略,Gulp 让简单的任务简单，复杂的任务可管理
- 快速构建 利用 node.js 流的威力,你可以快速构建项目并减少频繁的 IO 操作
- 高质量的插件 Gulp 严格的插件指南确保插件如你期望的那样简洁地工作
- 易于学习 通过最少的 API，掌握 gulp 毫不费力，构建工作尽在掌握

3. 安装gulp

```
npm install --g gulp-cli
npm install --save-dev gulp
```

4. 异步任务和组合任务

gulpfile.js

```
const fs = require('fs');
const through = require('through2');
const { series, parallel } = require('gulp');
function callbackTask(done) {
    setTimeout(() => {
        console.log('callbackTask');
        done();
    }, 1000);
}
function promiseTask() {
    return new Promise((resolve) => {
        setTimeout(() => {
            console.log('promiseTask');
            resolve();
        }, 1000);
    });
}
async function asyncTask() {
    await new Promise((resolve) => {
        setTimeout(() => {
            resolve();
        }, 1000);
    });
    console.log('asyncTask');
}
function streamTask() {
    return fs.createReadStream('input.txt')
        .pipe(through((chunk, encoding, next) => {
            setTimeout(() => {
                next(null, chunk);
            }, 1000);
        }))
        .pipe(fs.createWriteStream('output.txt'))
}

const parallelTask = parallel(callbackTask, promiseTask, asyncTask, streamTask);
const seriesTask = series(callbackTask, promiseTask, asyncTask, streamTask);
exports.callback = callbackTask
exports.promise = promiseTask
exports.async = asyncTask
exports.stream = streamTask
exports.parallel = parallelTask
exports.series = seriesTask
```

5. gulp核心API

- `gulp.src()`方法正是用来获取流的
- 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
- `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
- options 为可选参数

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

- `gulp.dest()`是用来向硬盘写入文件的
    - path 为写入文件的路径
    - options 为一个可选的参数对象
- `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
- `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
- 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

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

gulpfile.js

```
const { src, dest } = require('gulp');
function copyTask() {
    console.log('执行拷贝任务');
    return src('src/**/*.js').pipe(dest('dist'));
}
exports.default = copyTask;
```

6. gulp实战

```
cnpm install @babel/core @babel/preset-env browser-sync gulp gulp-babel gulp-clean gulp-clean-css gulp-ejs gulp-htmlmin gulp-if gulp-imagemin gulp-less gulp-load-plugins gulp-uglify gulp-useref gulp-watch map-stream bootstrap jquery --save
```

```
const { src, dest } = require('gulp');
const less = require('gulp-less');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

exports.styles = styles;
```

```
const { src, dest } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

+const scripts = () => {
+    return src("src/scripts/*.js", { base: 'src' })
+        .pipe(babel({
+            presets: ["@babel/preset-env"]
+        }))
+        .pipe(dest('dist'))
+}

exports.styles = styles;
+exports.scripts = scripts;
```

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><%=title%>title>
        <link rel="stylesheet" href="node_modules/bootstrap/dist/css/bootstrap.css"/>
    <link rel="stylesheet" href="styles/main.css"/>
head>
<body>
    <button class="btn btn-danger">按钮button>
    <img src="assets/images/circle.svg"/>
    <img src="rect.svg"/>
    <script src="node_modules/jquery/dist/jquery.js">script>
    [redacted]
```

src\assets\images\circle.svg

```

<svg width="100" height="100" version="1.1" xmlns="http://www.w3.org/2000/svg">
 <circle cx="50" cy="50" r="50"  fill="red"/>
svg>
```

static\rect.svg

```

<svg width="100" height="100" version="1.1" xmlns="http://www.w3.org/2000/svg">
 <rect  width="100" height="100" style="fill:red;"/>
svg>
```

```
const { src, dest, parallel } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const ejs = require('gulp-ejs');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

+const html = () => {
+    return src("src/*.html", { base: 'src' })
+        .pipe(ejs({ title: 'gulp' }))
+        .pipe(dest('dist'))
+}
exports.styles = styles;
exports.scripts = scripts;
+exports.html = html;
```

```
const { src, dest, parallel } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const ejs = require('gulp-ejs');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}
+const compile = parallel(styles, scripts, html);
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
+exports.compile = compile;
```

```
npm install gulp-imagemin --save-dev
npm install imagemin-jpegtran imagemin-svgo imagemin-gifsicle imagemin-optipng --save-dev
```

```
const { src, dest, parallel } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const ejs = require('gulp-ejs');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}

+const images = async () => {
+    let imagemin = await import('gulp-imagemin');
+    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
+        .pipe(imagemin.default())
+        .pipe(dest('dist'))
+}

const compile = parallel(styles, scripts, html);
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
+exports.images = images;
```

```
const { src, dest, parallel } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const ejs = require('gulp-ejs');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}

const images = async () => {
    let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

+const static = async () => {
+    return src("static/**", { base: 'static' })
+        .pipe(dest('dist'))
+}

const compile = parallel(styles, scripts, html);
+const build = parallel(compile, static)
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
exports.images = images;
+exports.static = static;
+exports.build = build;
```

```
const { src, dest, parallel, series } = require('gulp');
const less = require('gulp-less');
const babel = require('gulp-babel');
const ejs = require('gulp-ejs');
+const gulpClean = require('gulp-clean');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}

const images = async () => {
    let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

const static = async () => {
    return src("static/**", { base: 'static' })
        .pipe(dest('dist'))
}
+const clean =  () => {
+    return src("dist/**", { read: false })
+        .pipe(gulpClean())
+}
const compile = parallel(styles, scripts, html);
+const build = series(clean, parallel(compile, static));
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
exports.images = images;
exports.static = static;
exports.build = build;
+exports.clean = clean;
```

```
const { src, dest, parallel, series } = require('gulp');
-const less = require('gulp-less');
-const babel = require('gulp-babel');
-const ejs = require('gulp-ejs');
-const gulpClean = require('gulp-clean');
+const plugins = require('gulp-load-plugins')();
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
+       .pipe(plugins.less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
+       .pipe(plugins.babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
+       .pipe(plugins.ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}

const images = async () => {
     let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

const static = async () => {
    return src("static/**", { base: 'static' })
        .pipe(dest('dist'))
}
const clean =  () => {
    return src("dist/**", { read: false })
+       .pipe(plugins.clean())
}
const compile = parallel(styles, scripts, html);
const build = series(clean, parallel(compile, static));
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
exports.images = images;
exports.static = static;
exports.build = build;
exports.clean = clean;
```

```
const { src, dest, parallel, series } = require('gulp');
const plugins = require('gulp-load-plugins')();
+const browserSync = require('browser-sync');
+const path = require('path');
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(plugins.less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(plugins.babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(plugins.ejs({ title: 'gulp' }))
        .pipe(dest('dist'))
}

const images = async () => {
    let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

const static = async () => {
    return src("static/**", { base: 'static' })
        .pipe(dest('dist'))
}
const clean = () => {
    return src("dist/**", { read: false })
        .pipe(plugins.clean())
}
+const serve = () => {
+    return browserSync.create().init({
+        notify: false,
+        server: {
+            baseDir: 'dist',
+            routes: {
+                '/node_modules': path.resolve('node_modules')
+            }
+        }
+    });
+}
const compile = parallel(styles, scripts, html);
const build = series(clean, parallel(compile, static));
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
exports.images = images;
exports.static = static;
exports.build = build;
exports.clean = clean;
+exports.serve = serve;
```

src\scripts\main.js

```
let sum = (a, b) => a + b;
sum(1, 3)
+$(() => {
+    console.log('jquery');
+});
```

```
const { src, dest, parallel, series, watch } = require('gulp');
const plugins = require('gulp-load-plugins')();
const browserSync = require('browser-sync');
const path = require('path');
+const browserServer = browserSync.create();
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(plugins.less())
        .pipe(dest('dist'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(plugins.babel({
            presets: ["@babel/preset-env"]
        }))
        .pipe(dest('dist'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
+       .pipe(plugins.ejs({ title: 'gulp' }, { cache: false }))
        .pipe(dest('dist'))
}

const images = async () => {
    let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

const static = async () => {
    return src("static/**", { base: 'static' })
        .pipe(dest('dist'))
}
const clean = () => {
    return src("dist/**", { read: false })
        .pipe(plugins.clean())
}
const serve = () => {
+   watch("src/styles/*.less", styles);
+   watch("src/scripts/*.js", scripts);
+   watch("src/*.html", html);
+   watch([
+       "src/assets/images/**/*.@(jpg|png|gif|svg)",
+       "static/**"
+   ], browserServer.reload);
    return browserServer.init({
        notify: false,
+       files: ['dist/**'],
        server: {
+           baseDir: ['dist', 'src', 'static'],
            routes: {
                '/node_modules': path.resolve('node_modules')
            }
        }
    });
}
const compile = parallel(styles, scripts, html);
+const build = series(clean, parallel(compile, images, static));
+const dev = series(clean, compile, serve);
exports.styles = styles;
exports.scripts = scripts;
exports.html = html;
exports.compile = compile;
exports.images = images;
exports.static = static;
exports.clean = clean;
exports.serve = serve;
+exports.build = build;
+exports.dev = dev;
```

src\index.html

```

+

+

    按钮

+

+

const { src, dest, parallel, series, watch } = require('gulp');
const plugins = require('gulp-load-plugins')();
const browserSync = require('browser-sync');
const path = require('path');
const browserServer = browserSync.create();
const styles = () => {
    return src("src/styles/*.less", { base: 'src' })
        .pipe(plugins.less())
+       .pipe(dest('temp'))
}

const scripts = () => {
    return src("src/scripts/*.js", { base: 'src' })
        .pipe(plugins.babel({
            presets: ["@babel/preset-env"]
        }))
+       .pipe(dest('temp'))
}

const html = () => {
    return src("src/*.html", { base: 'src' })
        .pipe(plugins.ejs({ title: 'gulp' }, { cache: false }))
+       .pipe(dest('temp'))
}

const images = async () => {
    let imagemin = await import('gulp-imagemin');
    return src("src/assets/images/**/*.@(jpg|png|gif|svg)", { base: 'src' })
        .pipe(imagemin.default())
        .pipe(dest('dist'))
}

const static = async () => {
    return src("static/**", { base: 'static' })
        .pipe(dest('dist'))
}
const clean = () => {
+   return src(["dist/**", "temp/**"], { read: false })
+       .pipe(plugins.clean({ allowEmpty: true }));
}
const serve = () => {
    watch("src/styles/*.less", styles);
    watch("src/scripts/*.js", scripts);
    watch("src/*.html", html);
    watch([
        "src/assets/images/**/*.@(jpg|png|gif|svg)",
        "static/**"
    ], browserServer.reload);
    return browserServer.init({
        notify: false,
        files: ['dist/**'],
        server: {
+           baseDir: ['temp', 'src', 'static'],
            routes: {
                '/node_modules': path.resolve('node_modules')
            }
        }
    });
}

+const concat = () => {
+    return src('temp/*.html', { base: 'temp' })
+        .pipe(plugins.useref({
+            searchPath: ['temp', '.']
+        }))
+        .pipe(plugins.if('*.html', plugins.htmlmin({
+            collapseWhitespace: true,
+            minifyCSS: true,
+            minifyJS: true
+        })))
+        .pipe(plugins.if('*.js', plugins.uglify()))
+        .pipe(plugins.if('*.css', plugins.cleanCss()))
+        .pipe(dest('dist'));
+}

const compile = parallel(styles, scripts, html);
+const build = series(clean, parallel(series(compile, concat), images, static));
const dev = series(clean, compile, serve);
-exports.styles = styles;
-exports.scripts = scripts;
-exports.html = html;
-exports.compile = compile;
-exports.images = images;
-exports.static = static;
-exports.serve = serve;
-exports.concat = concat;
-exports.clean = clean;
-exports.build = build;
-exports.dev = dev;

+module.exports = {
+    clean,
+    build,
+    dev
+}
```

7. 参考知识

- gulp内部使用了node-glob模块来实现其文件匹配功能

匹配符 说明 * 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 ** 匹配路径中的0个或多个目录及其子目录 ? 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 !(pattern1 pattern2 pattern3) 匹配任何与括号中给定的任一模式都不匹配的 ?(pattern1 pattern2 pattern3) 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(pattern1 pattern2 pattern3)? +(pattern1 patter2n pattern3) 匹配括号中给定的任一模式至少1次，类似于js正则中的(pattern1 pattern2 pattern3)+ (pattern1 pattern2 pattern3) 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(pattern1 pattern2 pattern3)* @(pattern1 pattern2 pattern3) 匹配括号中给定的任一模式1次，类似于js正则中的(pattern1 pattern2 pattern3)

glob 匹配 * 能匹配 a.js,x.y,abc,abc/,但不能匹配a/b.js

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 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