Gulp 的使用

作者：xfy196
原文链接：https://ld246.com/article/1586674362954
来源网站：链滴

Gulp是一个前端自动化构建工具，基于流的的构建系统，主要优点是速度，效率和简化

Gulp可以干什么?

开发服务器，Sass,Less或Stylus文件的预处理器，处理JavaScript的自动化脚本，代码优化工具，压，编译或移动文件

Gulp的官网 https://www.gulpjs.com.cn/

gulp类似的工具 : grunt webpack

gulp是基于nodejs的， gulp和所有插件都通过JavaScript编写并依托Node.js平台

1.gulp优点:

1. 易于使用
2. 构建快速
3. 插件高质
4. 易于学习

2.gulp的安装

全局安装

npm install gulp -g
npm install gulp@版本号 -g

执行gulp -v查看gulp的版本号

局部安装

在本地下载
npm install gulp --save-dev

3.gulp api

1.gulp.task(name, fn) 构建任务

name任务名 string

任务名为default： 执行 ---> gulp

任务名为minJs : 执行 ---> gulp 任务名

2.gulp.src(globs) 读取文件

- 通配符路径匹配

1. “src/a.js”

指定具体文件

2. “*” :匹配所有文件

例： src/*.js (包含src下的所有的js文件)

3. “**” ：匹配0个或多个子文件夹

例： src/**/*.js（包含src的0个多个子文件夹下的js文件）

4. “{}” :匹配对个属性

例：src/{a,b}.js(包含a.js和b.js文件) src/*.{jpg,png,gif}(src下的所有jpg/png/gif文件)；

5."!" : 排除文件

例：!src/a.js(不包含src下的a.js文件)；

3.gulp.dest(输出的路径) 输出文件

4.gulp.series(task1, task2....) 设置任务执行顺序 串行执行

5.gulp.parallel(task1, task2,....) 设置任务执行顺序 并行执行

6.gulp.watch(globs, gulp.series/gulp.parallel()task1,task2,...) 监听文件变化，执行任务

gulp插件

插件	说明
http-proxy-middleware	服务器代理
gulp-sass	编译scss
gulp-less	编译less

gulp-concat	自动添加前缀
gulp-concat	合并文件
gulp-clean-css	缩css
gulp-webserver	起服务
browser-sync	起服务
gulp-htmlmin	压缩html
gulp-imagemin	压缩图片
gulp-uglify	压缩js 不支持ES6
gulp-babel	ES6转ES5

编译scss与sass

```
# 安装gulp-sass
npm i gulp-sass --save-dev
npm i node-sass --save-dev

const sass = require("gulp-sass");
sass.compiler    = require('node-sass');
// sass的使用需要下载gulp-sass和node-sass
gulp.task("sass", async()=>{
    gulp.src(["./src/sass/**/*.scss"]) //指定sass的文件地址
    .pipe(sass()) // 调用sass处理的函数
    .pipe(gulp.dest("./dist/css/")); // 输出的地址
});
```

压缩css

```
# 安装gulp-csssnano
npm i gulp-cssnano --save-dev

const cssnano = require("gulp-cssnano")
gulp.task("cssnano", async()=>{
    gulp.src(["./src/css/**/*.css"])
    .pipe(cssnano()) // 调用cssnano的函数压缩css代码
    .pipe(gulp.dest("./dist/css/"));
});
```

起服务 gulp-connect

```
# 安装gulp-connect
npm i gulp-connect --save-dev

const connect = require("gulp-connect");
let connect_options = {
    root : "./dist/",  // 服务的根目录
    port : 3000,
    livereload : true
}
```

```
gulp.task("connect", async()=>{
    connect.server(connect_options);  // 启动服务
})
```

压缩html

```
# 安装gulp-htmlmin
npm i gulp-htmlmin --save-dev

const htmlMin = require("gulp-htmlmin");
gulp.task("htmlmin", async()=>{

    gulp.src(["./src/html/**/*.html"])
    .pipe(htmlMin()) // 压缩html代码
    .pipe(gulp.dest("./dist/"));
})
```

gulp-htmlmin的参数设置

```
{
    removeComments: true,//清除HTML注释
    collapseWhitespace: true,//压缩HTML
    collapseBooleanAttributes: true,//省略布尔属性的值 <input checked="true"/> ==> <input /

    removeEmptyAttributes: true,//删除所有空格作属性值 <input id="" /> ==> <input />
    removeScriptTypeAttributes: true,//删除<script>的type="text/javascript"
    removeStyleLinkTypeAttributes: true,//删除<style>和<link>的type="text/css"

}
```

压缩图片

```
# 安装gulp-imagemin
npm i gulp-imagemin --save-dev

const imagemin = require("gulp-imagemin");
gulp.task("imagemin", async () => {

    gulp.src(["./src/images/**/*.{png,jpg,jpeg,gif}"])
        .pipe(imagemin({
            optimizationLevel: 5, //类型：Number  默认：3  取值范围：0-7（优化等级）
            progressive: true, //类型：Boolean 默认：false 无损压缩jpg图片
            interlaced: true, //类型：Boolean 默认：false 隔行扫描gif进行渲染
            multipass: true //类型：Boolean 默认：false 多次优化svg直到完全优化
        })).pipe(gulp.dest("./dist/images/"));
})
```

压缩js

```
# 安装 uglify
npm i gulp-uglify --save-dev
```

```
const uglify = require("gulp-uglify");

gulp.task("uglify", async()=>{
    gulp.src(["./src/js/**/*js"])
    .pipe(uglify()) // 压缩js
    .pipe(gulp.dest("./dist/js/"));
})
```

ugilify的参数

```
{
    mangle: true,//类型：Boolean 默认：true 是否修改变量名
    mangle: {except: ['require' ,'exports' ,'module' ,'$']}//排除混淆关键字
    compress: true,//类型：Boolean 默认：true 是否完全压缩
    preserveComments: 'all' //保留所有注释
}
```

编译ES6

```
# 安装 gulp-babel
# Babel 7
$ npm install --save-dev gulp-babel @babel/core @babel/preset-env

# Babel 6
$ npm install --save-dev gulp-babel@7 babel-core babel-preset-env

const babel = require("gulp-babel");

gulp.task("babel", async()=>{
    gulp.src(["./src/js/**.js"])
    .pipe(babel({
        presets : ["@babel/env"]
    })).pipe(gulp.dest("./dist/js/"));
})
```

代理

```
# 安装代理 http-proxy-middleware
npm i http-proxy-middleware --save-dev

const proxy = require("http-proxy-middleware");
// 需要配合服务器插件使用
let connect_options = {
    root: "./dist/",  // 服务的根目录
    port: 3000,
    livereload: true,
    middleware : ()=>{
        return [
            proxy("/pdd",{
                target : "https://xxx.xxx.xxx.xxx/api", //api地址
                changeOrigin : true, // 是否改变同源
                pathRewrite : {
                    "/pdd" : ""  // 将多余的地址替换成
```

```
            }
        })
    ]
  }
}
```

开发环境

gulp.task('dev',gulp.series('任务名','任务名'))

线上环境

gulp.task('build',gulp.parallel('任务名','任务名'))