### 学习Gulp

1. gulp简介

gulp是一个前端自动化的工具【构建工具】，使用流(stream)来处理 内容。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。

1. 用途

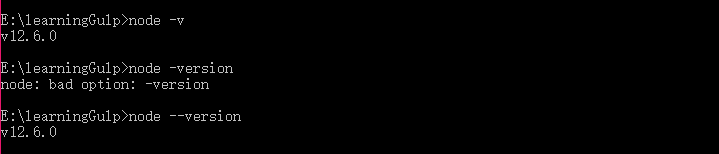
* 搭建web服务器
* 使用预处理器Sass，Less
* 压缩优化资源，可以压缩JS CSS Html 图片
* 自动将更新变化的代码实时显示在浏览器【前端实时化】
* ……

1. 下载node并安装
2. 下载地址

http://nodejs.cn/download/

1. 查看版本

node -v 或 node --version



1. 安装gulp
2. gulp官网

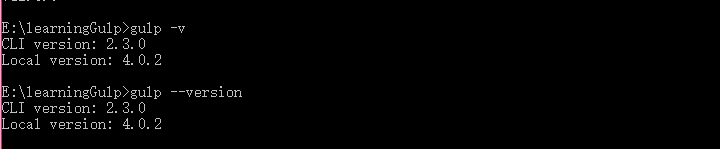
   https://www.gulpjs.com.cn/

1. 先全局安装(系统管理员cmd默认路径下全局安装)

   npm install -g gulp

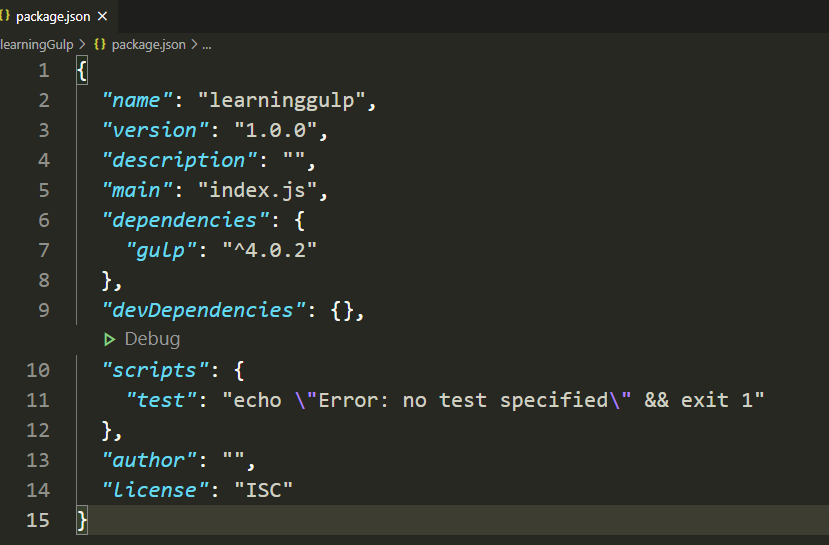
1. 查看版本

gulp -v 或 gulp --version



1. 切换到项目根目录，执行npm init 一路enter键初始化打包信息，生 成package.json文件

npm init



1. 然后局部安装(会在项目根目录中生成node\_modules文件夹和 package-lock.json文件)

npm install gulp --save-dev

1. 指定版本安装gulp

npm install gulp@3.9.1 --save-dev

1. 卸载gulp

npm uninstall gulp

1. 创建gulpfile.js文件
2. gulp模块的方法
3. task()

task方法用于定义具体的任务。它的第一个参数是任务名，第二个参数 是任务函数。

5.1.1 下面是一个简单的任务函数

gulp.task('greet', *function* () {

console.log('Hello world!');

});

5.1.2 task方法还可以指定按顺序运行的一组任务。

下面代码先指定build任务，它由css、js、imgs三个任务所组成， task方法会并发执行这三个任务。注意，由于每个任务都是异步调用，所以没有办法保证js任务的开始运行的时间，正是css任务运行结束。

gulp.task('build', ['css', 'js', 'imgs']);

5.1.3 各个任务严格按次序运行，可以把前一个任务写成后一个任务的 依赖模块。

下面代码表明，css任务依赖greet任务，所以css一定会在greet 运行完成后再运行。

gulp.task('css', ['greet'], *function* (){

    // Deal with CSS here

});

5.1.4 task方法的回调函数，还可以接受一个函数作为参数，这对执行 异步任务非常有用。

//执行shell命令

*var* exec = require('child\_process').exec;

gulp.task('jekyll', *function*(*cb*) {

    // build Jekyll

    exec('jekyll build', *function*(*err*) {

       if (*err*) return cb(*err*);//return error

        cb(); // finished task

    });

});

5.1.5 如果一个任务的名字为default，就表明它是“默认任务”，在 命令行直接输入gulp命令，就会运行该任务。

执行的时候，直接使用gulp，就会运行styles、jshint、 watch三个任务。

gulp.task('default', *function* () {

  // Your default task

});

// 或者

gulp.task('default', ['styles', 'jshint', 'watch']);

1. src()

gulp模块的src方法，用于产生数据流。它的参数表示所要处理的文件， 这些指定的文件会转换成数据流。参数的写法一般有以下几种形式。

5.2.1 指定确切的文件名

  gulp.src('js/app.js')

5.2.2 某个目录所有后缀名为js的文件

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

5.2.3 某个目录及其所有子目录中的所有后缀名为js的文件

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

5.2.4 除了js/app.js以外的所有文件

gulp.src('!js/app.js')

5.2.5 匹配项目根目录下，所有后缀名为js或css的文件

gulp.src('\*.+(js  css)')

5.2.6 参数还可以是一个数组，用来指定多个成员

gulp.src(['js/\*\*/\*.js', '!js/\*\*/\*.min.js'])

1. dest()

dest方法将管道的输出写入文件，同时将这些输出继续输出，所以可以 依次调用多次dest方法，将输出写入多个目录。如果有目录不存在，将 会被新建。

5.3.1 简单的一个例子

gulp.task('build',*function*(){

 return gulp.src('./client/templates/\*.jade')

  .pipe(jade())

  .pipe(gulp.dest('./build/templates'))

  .pipe(minify())

  .pipe(gulp.dest('./build/minified\_templates'));

})

5.3.2 dest方法还可以接受第二个参数，表示配置对象。

配置对象有两个字段。cwd字段指定写入路径的基准目 录，默认是 当前目录；mode字段指定写入文件的权限， 默认是0777。

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

     gulp.src('js/\*.css')

     .pipe(gulp.dest('build', {

        cwd: './app',

        mode: '0644'

     }))

})

1. watch()

watch方法用于指定需要监视的文件。一旦这些文件发生变动，就运行 指定任务。

5.4.1 简单的一个例子

下面代码指定，一旦templates目录中的模板文件发生变化，就运 行build任务。

gulp.task('watch', *function* () {

   gulp.watch('templates/\*.tmpl.html', ['build']);

});

5.4.2 watch方法也可以用回调函数，代替指定的任务

gulp.watch('templates/\*.tmpl.html', *function* (*event*) {

   console.log('Event type: ' + *event*.type);

   console.log('Event path: ' + *event*.path);

});

5.4.3 另一种写法是watch方法所监控的文件发生变化时（修改、增加、删除文件），会触发change事件。可以对change事件指定回调函数

*var* watcher = gulp.watch('templates/\*.tmpl.html', ['build']);

watcher.on('change', *function* (*event*) {

   console.log('Event type: ' + *event*.type);

   console.log('Event path: ' + *event*.path);

});

5.4.5 除了change事件，watch方法还可能触发以下事件

1. end： 回调函数运行完毕时触发
2. error： 发生错误时触发
3. ready： 当开始监听文件时触发
4. nomatch： 没有匹配的监听文件时触发

5.4.6 watcher对象还包含其他一些方法

1. watcher.end()： 停止watcher对象，不会再调用任务或回调函数
2. watcher.files()： 返回watcher对象监视的文件
3. watcher.add(glob)： 增加所要监视的文件，它还可以附件第二个参数，表示回调函数
4. watcher.remove(filepath)： 从watcher对象中移走一个监视的文件
5. gulp-load-plugins 模块
6. 一般情况下，gulpfile.js中的模块需要一个个加载

除了gulp模块以外，还加载另外三个模块。

这种一一加载的写法，比较麻烦

*var*  gulp = require('gulp');

*var*  jshint = require('gulp-jshint');

*var*  uglify = require('gulp-uglify');

*var*  concat = require('gulp-concat');

gulp.task('js', *function* () {

   return gulp.src('js/\*.js')

      .pipe(jshint())

      .pipe(jshint.reporter('default'))

      .pipe(uglify())

      .pipe(concat('app.js'))

      .pipe(gulp.dest('build'));

});

1. 使用gulp-load-plugins模块，可以加载package.json文件 中所有的gulp模块。

上面的代码用gulp-load-plugins模块改写，就是下面这样

*var* gulp = require('gulp');

*var* gulpLoadPlugins = require('gulp-load-plugins');

*var* 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'));

});

1. Gulp-livereload模块
2. gulp-livereload模块用于自动刷新浏览器，反映出源码的最新变化。它 除了模块以外，还需要在浏览器中安装插件，用来配合源码变化。

下面代码监视less文件，一旦编译完成，就自动刷新浏览器。

1. 必备插件
2. gulp-htmlmin 这个插件是用来压缩 HTML
3. gulp-imagemin 除了能压缩常见的图片格式，还能压缩 SVG
4. gulp-clean-css 压缩 CSS
5. gulp-uglify 专业压缩 Javascript
6. gulp-concat 上面几个都是压缩，这插件是管合并的
7. 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
8. gulp-rename 修改文件名称。比如有时我们需要把app.js改成 app.min.js
9. gulp-util 最基础的工具，但俺只用来打日志...
10. gulp-sass / gulp-less
11. gulp-jshint JavaScript 代码校验
12. gulp-load-plugins 帮忙偷懒用的，可以帮我们加载插件，不用require或者import...