gulp学习之你来我往 src和dest函数

作为一个前端构建工具，gulp无疑是我接触过的最简单易用的。其配置参数简单，并且学习起来也很容易。并且gulp使用的是nodejs的stream来操作数据的，所以运行效率也是比较高的。所以，之前没有使用过前端构建工具的同学，我个人还是建议学习一下gulp。

gulp的API也比较简单，总共就提供给我们四个函数：gulp.src、gulp.dest、gulp.task和gulp.watch。本章我们来讨论src和dest函数。

**gulp.src**

gulp底层是使用的nodejs的stream，首先获取到stream流，然后使用pipe()方法把流数据导入到你想让它去的地方。

而gulp.src就是来获取这个stream的。

gulp.src(globs[, options]);

globs参数类型可以是字符串也可以是数组。

例如

gulp.src(‘lib/onmpw.js’);

或者

gulp.src([‘lib/onmpw.js’,’lib/jiyi.js’]);

globs这个参数是支持简单的正则表达式的，下面来看几个常用的字符

\*匹配0个或多个字符，但是不会匹配路径分隔符。也就是说该字符的的作用范围是当前目录，不会匹配到下一级目录

gulp.src(‘lib/\*.js’);

匹配lib目录下所有的js文件，加入lib目录下有a.js，b.js，ab.js，bc.js和onmpw目录。能匹配到a.js，b.js，ab.js和bc.js。但是并不能匹配到onmpw目录下的js文件。

! 排除后面跟着的js文件。

gulp.src(‘lib/!b.js’)

这里除了lib中的b.js，其余的js文件都能匹配到。

\*\*匹配路径中的0个或多个目录及其子目录。该匹配符需要单独出现，左右都不能出现其他的字符。

gulp.src(‘lib/\*\*/\*.js’);

匹配lib目录下的所有js文件，包括其子目录下的js文件。这个就是和lib/\*.js的区别。

其中多种匹配模式可以使用数组来实现

gulp.src([‘lib/\*.js’,’!lib/b\*.js’,’lib/bc.js’]);

匹配lib目录下的所有的js文件（仅当前层级，不包括子目录），然而要排除掉以b开头的js文件。但是在b开头的js文件中不包括bc.js。

（当然，除了js文件。还可以读取其它多种类型的文件——css，html，jade等）。

除了上面，globs还支持很多种匹配方式，下面是列举的几种方式，我们就不一一举例子了。

[...] 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，类似js正则表达式中的用法

!(pattern|pattern|pattern) 匹配任何与括号中给定的任一模式都不匹配的

?(pattern|pattern|pattern) 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(pattern|pattern|pattern)?

+(pattern|pattern|pattern) 匹配括号中给定的任一模式至少1次，类似于js正则中的(pattern|pattern|pattern)+

\*(pattern|pattern|pattern) 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(pattern|pattern|pattern)\*

@(pattern|pattern|pattern) 匹配括号中给定的任一模式1次，类似于js正则中的(pattern|pattern|pattern)

对于匹配方式更详细的知识，我们可以参考[node-glob syntax](https://github.com/isaacs/node-glob)。

options这个参数指定以什么类型读取文件的内容。

options.buffer 默认值为true，如果指定为false，则返回的文件内容为stream类型而不是buffer。当我们要读取的文件相当大的时候可以将该参数指定为false，这是很有效的方式。

由于这个参数经常性的用不到，所以这里不深入讨论。有兴趣的可以参考<http://gulpjs.org/API.html> 。

**gulp.dest**

要说src函数是读取文件，那么dest函数就是写文件。

gulp.dest(path[,options])

path是要写入的文件的路径，这里注意只能是路径，不能是具体的文件。

var gulp = require(‘gulp’);

gulp.src(‘lib/onmpw.js’)

.pipe(gulp.dest(‘dst’));

生成的文件路径为dst/onmpw.js。即使在dest中指定了文件名，也会将文件名按照目录来解析。

var gulp = require(‘gulp’);

gulp.src(‘lib/onmpw.js’)

.pipe(gulp.dest(‘dst/jiyi.js’));

生成的文件路径为dst/jiyi.js/onmpw.js。如果我们想改变写入的文件的名称该怎么办。可以使用gulp-rename插件。

在介绍src函数的时候，我们介绍了在其globs参数中我们可以传入通配符。对于通配符匹配到的文件dest函数是怎么处理的呢？下面我们举几个例子就明白了

var gulp = require(‘gulp’);

gulp.src(‘lib/\*\*/\*.js’)

.pipe(gulp.dest(‘dst’));

假设src中匹配到的文件为 lib/script/onmpw.js。则生成的文件路径为dst/script/onmpw.js。

var gulp = require(‘gulp’);

gulp.src(‘lib/script/\*.js’)

.pipe(gulp.dest(‘dst’));

假设src中匹配到的文件为 lib/script/onmpw.js。则生成的文件路径为dst/ onmpw.js。

通过上面两个例子大概可以得出结论就是，dest函数中生成的文件路径要从src中第一个通配符出现的地方开始。然而这种情况是可以改变的。可以在src函数中设置base选项改变这个位置。

var gulp = require(‘gulp’);

gulp.src(‘lib/script/\*.js’,{base:’lib’})

.pipe(gulp.dest(‘dst’));

假设匹配到的文件为 lib/script/onmpw.js 如果不设置base选项的话，生成的文件路径为dst/onmpw.js。现在我们设置了从lib位置的下一个位置开始，也就是从script开始。所以生成的文件路径为dst/script/onmpw.js。

当我们使用dest函数将src读取的文件流写入文件以后，文件流依然是可以继续流转的。

src和dest函数是成对出现的，单出现任意一个函数都不能完成任务。