前端静态资源版本更新与缓存之——通过gulp 在原html文件上自动化添加js、css版本号

原理

1. 修改js和css文件
2. 通过对js,css文件内容进行hash运算，生成一个文件的唯一hash字符串(如果文件修改则hash号会发生变化)
3. 替换html中的js,css文件名，生成一个带版本号的文件名

方案

现在网上的方案都是生成一个新的dist目录，里面包含了要发布的html,js,css等文件。但是在实际的公司的项目中，会有情况不能生成新的HTML进行发布，需要在原来的HTML文件上进行js ,css版本的替换. 这里分享下我在实际项目中通过改动插件然后在原目录结构下进行版本的控制方案。

原html文件代码

```
<link rel="stylesheet" href="../css/default.css">
<script src="../js/app.js"></script>
```

预期效果：在原目录结构下html文件代码

```
<link rel="stylesheet" href="../css/default.css?v=5a636d79c4">
<script src="../js/app.js?v=3a0d844594"></script>
```

1：安装gulp和gulp插件

执行：

```
npm install --save-dev gulp
npm install --save-dev gulp-rev
npm install --save-dev gulp-rev-collector
npm install --save-dev run-sequence
```

2：编写gulpfile.js

```
//引入gulp和gulp插件
var gulp = require('gulp'),
    runSequence = require('run-sequence'),
    rev = require('gulp-rev'),
    revCollector = require('gulp-rev-collector');

//定义css、js源文件路径
var cssSrc = 'css/*.css',
    jsSrc = 'js/*.js';


//CSS生成文件hash编码并生成 rev-manifest.json文件名对照映射
gulp.task('revCss', function(){
    return gulp.src(cssSrc)
        .pipe(rev())
        .pipe(rev.manifest())
        .pipe(gulp.dest('rev/css'));
});


//js生成文件hash编码并生成 rev-manifest.json文件名对照映射
gulp.task('revJs', function(){
```

```
    return gulp.src(jsSrc)
        .pipe(rev())
        .pipe(rev.manifest())
        .pipe(gulp.dest('rev/js'));
});


//Html替换css、js文件版本
gulp.task('revHtml', function () {
    return gulp.src(['rev/**/*.json', 'View/*.html'])
        .pipe(revCollector())
        .pipe(gulp.dest('View'));
});


//开发构建
gulp.task('dev', function (done) {
    condition = false;
    runSequence(
        ['revCss'],
        ['revJs'],
        ['revHtml'],
        done);
});


gulp.task('default', ['dev']);
```

执行gulp命令后的效果

```
//rev目录下生成了manifest.json对应文件
{
  "default.css": "default-803a7fe4ae.css"
}


<link rel="stylesheet" href="../css/default-803a7fe4ae.css">
<script src="../js/app-3a0d844594.js"></script>
```

很显然这不是我们需要的效果

3.更改gulp-rev和gulp-rev-collector

```
打开node_modules\gulp-rev\index.js
第144行 manifest[originalFile] = revisionedFile;
更新为: manifest[originalFile] = originalFile + '?v=' + file.revHash;


打开nodemodules\gulp-rev\nodemodules\rev-path\index.js
10行 return filename + '-' + hash + ext;
更新为: return filename + ext;


打开node_modules\gulp-rev-collector\index.js
31行if ( !_.isString(json[key]) || path.basename(json[key]).replace(new RegExp( opts.revSuffix ), '' ) !==  pat
h.basename(key) ) {
更新为: if ( !_.isString(json[key]) || path.basename(json[key]).split('?')[0] !== path.basename(key) ) {
```

再执行gulp命令，得到的结果如下(效果正确):

```
<link rel="stylesheet" href="../css/default.css?v=803a7fe4ae">
<script src="../js/app.js?v=3a0d844594"></script>
```

但是假如我们更改了css和js后，再执行gulp命令，得到的结果会如下:

```
<link rel="stylesheet" href="../css/default.css?v=33379df310?v=803a7fe4ae">
<script src="../js/app.js?v=3a0d844594?v=3a0d844594"></script>
```

有没有发现，会在版本号后面再添加一个版本号，因为gulp只替换了原来文件名，这样又不符合预期效果了，所以我们想到，还需要修改插件的替换正则表达式。

4.继续更改gulp-rev-collector

```
打开node_modules\gulp-rev-collector\index.js
第107行 regexp: new RegExp( '([\/\\\\\'"])' + pattern, 'g' ),
更新为: regexp: new RegExp( '([\/\\\\\'"])' + pattern+'(\\?v=\\w{10})?', 'g' ),
```

现在你不管执行多少遍gulp命令，得到的html效果都是

```
<link rel="stylesheet" href="../css/default.css?v=5a636d79c4">
<script src="../js/app.js?v=3a0d844594"></script>
```

附上改过后的node_modules文件