Gulp - Build Automation Tool

Gulp 란?

- Gulp는 Sass(Css Extension language) 컴파일이나 웹 리소스 최적화 같이 번거로운 작업들을 자동화 하는 Build Automation Tool
- Stream 기반으로 압축, 병합, 에러 처리 등의 여러 플러그인의 결과를 이어받아 최종 결과물을 출력할 수 있다.

Gulp **설치 & 시작하기**

- Gulp를 설치하기 위해서는 node.js가 pc에 설치되어 있어야 한다.

노드 설치 후 아래 절차로 gulp **환경세팅**

1. `npm i gulp --global` 를 이용하여 Gulp 를 설치한다.
2. `npm init` 명령어를 이용하여 package.json 파일을 생성한다
    - name, description, commands... etc
3. 프로젝트의 devDependencies에 Gulp를 추가하기 위해서, `npm i gulp --save-dev` 를 입력한다.

- gulpfile.js 파일을 프로젝트 루트에 위치시킨다.
- gulpfile.js 파일에 다음의 내용을 추가해보자.

```
/* File: gulpfile.js */
var gulp  = require('gulp');
var gutil = require('gulp-util');

// create a default task and just log a message
gulp.task('default', function() {
 return gutil.log('Gulp is running!')
});
```

- 추가 후 gulp를 치면, `Gulp is running` 이라는 콘솔이 찍혀야 구성이 제대로 된 것 이다.

Gulp **파일 구성을 위한 필수** API **4가지**

- gulp.task() - 작업 지정 및 수행
- gulp.src() - 작업 후 파일이 반환될 경로
- gulp.dest() - 최종 작업후 결과물 경로
- gulp.watch() - 지정된 소스의 변경시 수행할 작업 지정

gulp.task()

- 특정 작업을 지정하고 수행하는 명령어

```
gulp.task('taskname', function() {
 // 실행할 로직 추가
});
```

task() API 형식

```
gulp.task(name, [deps], fn)
```

- `name` : task 의 명칭. 이름 중간에 스페이스는 허용되지 않는다.
- `deps` : 해당 task 를 수행하기 전에 실행 및 완료되어야 할 tasks 의 목록. array 형태며 선택 사항이다.
- `fn` : task 의 실제 수행할 로직이 들어가는 function. 선택 사항이다.

```
gulp.task('task1', function() {
  var a = 1,
      b = 2;
  console.log(a+b); // 3
  console.log("task1 실행완료");
});

gulp.task('task2', ['task1'], function() {
  console.log("task2 실행");
});
```

```
// function 을 제외하고, build 작업을 수행하기 위해 미리 처리되어야 할 작업들만 명시
// 주의할 점은 배열의 작업들이 순차적으로 처리되는 게 아니라 병행으로 처리됨
gulp.task('build', ['array', 'of', 'task', 'names']);
```

gulp task 비동기 처리

- task 를 순차적으로 처리하는 것이 아니라, 아래와 같이 async 하게 처리가 가능할 수 있다.

```
// run a command in a shell
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
 });
});

// use an async result in a pipe
gulp.task('somename', function(cb) {
 getFilesAsync(function(err, res) {
   if (err) return cb(err);
   var stream = gulp.src(res)
     .pipe(minify())
     .pipe(gulp.dest('build'))
     .on('end', cb); // 위 jekyll 에서 작업한 결과가 콜백으로 반환되면 somename 이 수행된다.
 });
});
```

- 아래는 stream 방식으로 처리하는 방법이다.

```
gulp.task('somename', function() {
 var stream = gulp.src('client/**/*.js')
   .pipe(minify())
   .pipe(gulp.dest('build'));
 return stream;
});
```

gulp.src()

- 해당 위치의 파일들이 설정한 로직을 거쳐 Vinyl 파일의 Stream 형태로 반환된다.
- `.pipe` 를 이용해서 결과 값을 다른 플러그인으로 넘겨줄 수 있다.

```
gulp.src('client/templates/*.jade')
 .pipe(jade())
 .pipe(minify())
 .pipe(gulp.dest('build/minified_templates'));
```

src() API 형식

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

- `globs` : 1개 또는 여러 개의 파일을 String 또는 Array 로 지정이 가능하다. 파일 지정 형식에 관해서는 아래를 참고하자.

```
// 파일 1개 지정시
gulp.src('client/*.js')

// 배열을 이용한 여러 개의 파일 지정 및 규칙
gulp.src(['client/*.js', '!client/b*.js', 'client/bad.js'])
 // * 를 이용하여 복수 개의 파일 지정 가능.
 // ! 를 이용하여 해당 파일 배제
```

- `options` : 설정한 옵션은 glob-stream 을 이용하여 node-glob 에 넘겨진다. ??
- `options.buffer` : 값을 `false` 로 놓으면 버퍼 형식이 아니 스트림 형식으로 결과를 낸다.
- `options.read` : 'false' 로 설정하면 파일 읽기가 안되고, `file.contents` 값이 null 이 된다.

gulp.dest()

- 최종 결과값이 출력될 파일 위치를 지정한다. 해당 위치에 폴더가 없는 경우에는 새로 생성한다.

```
gulp.task('copyHtml', function() {
 // copy any html files in source/ to public/
 gulp.src('source/*.html').pipe(gulp.dest('public'));
});
```

gulp.watch()

- 첫 번째 지정한 인자 (파일) 에 변경이 일어나면 두 번째 인자 (작업) 의 작업이 수행된다.

```
// `source/javscript`의 밑에 위치한 js 파일이 변경되면, `jshint` 태스크가 수행된다.
gulp.watch('source/javascript/**/*.js', ['jshint']);
// API 수행 결과로는 change 이벤트를 발생시키는 EventEmitter 를 반환한다.
```

watch() API 형식

```
gulp.watch(glob, [cb])
```

- `glob` : String 또는 Array 로 변화를 감시할 파일들을 지정한다.
- `cb(event)` : 지정한 파일의 변화가 있을 때 마다 호출될 함수.

```
gulp.watch('js/**/*.js', function(event) {
 console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');
 // event type : added, changed, delted, renamed 중 어떤 변화가 일어났는지
 // event path : change 이벤트를 발생시킨 파일의 경로
});
```

Gulp 유용한 플러그인 목록

- Gulp Utility
- Gulp Uglify
- Gulp Concat
- Sourcemap
- Gulp HTML min
- Gulp CSS nano
- Gulp Uncss
- Gulp Image-min

Gulp Utility - gulp-util

- gulp 사용에 필요할만한 유틸 라이브러리 집합
- log 결과에 색깔을 주는 기능 등 빌드 작업과 관련한 기능들이 많다.

```
gutil.log(gutil.colors.magenta('123'));
```

Clean files - gulp-del

- 해당 위치의 파일을 삭제

```
var del = require('del');

del(['tmp/*.js', '!tmp/unicorn.js']).then(function(path) {
    console.log('Deleted files and folders:\n', paths.join('\n'));
});
```

Run synchronously - gulp-runsequence

- 태스크들을 병행으로 처리하지 않고, 정한 순서대로 처리한다.

```
// This will run in this order:
// * build-clean
// * build-scripts and build-styles in parallel
// * build-html
// * Finally call the callback function
var runSequence = require('run-sequence');

gulp.task('build', function(callback) {
  runSequence('build-clean',
              ['build-scripts', 'build-styles'],
              'build-html',
              callback);
});
```

gulp-plumber

- Gulp 실행 시 발생하는 처리에 대한 편의 기능 제공하는 플러그인
- Gulp 를 사용할 때 까다로운 점은 에러 처리를 하기 위한 로직을 별도로 추가해야 한다.
- 에러 핸들러를 추가하는 것 이외에도 Plumber 를 이용하여 에러를 catch 하고 pipe 가 깨지는 것을 막을 수 있다.

Gulp Optimization & Minification Plugin List

JS minification - gulp-uglify

- 자바스크립트 파일의 바이트를 최소화

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

// ...
```

JS concatenation - gulp-concat

- 자바스크립트 파일을 압축하는 것 이외에도, 여러 개의 js 파일을 한 개로 병합하면 왕복 네트워크 요청을 줄여 성능이 개선될 수 있다.

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

// 아래 배열 요소 순서대로 file1, file2, file3 을 병합한다.
gulp.task('scripts', function() {
  return gulp.src(['./lib/file1.js', './lib/file2.js', './lib/file3.js'])
    .pipe(concat('all.js'))
    .pipe(gulp.dest('./dist/'));
});
```

- concat 사용에 있어서 sourcemap 플러그인은 선택이 아닌 필수다

```
var gulp = require('gulp');
var concat = require('gulp-concat');
var sourcemaps = require('gulp-sourcemaps');

gulp.task('javascript', function() {
  return gulp.src('src/**/*.js')
    .pipe(sourcemaps.init())
      .pipe(concat('all.js'))
    .pipe(sourcemaps.write())
    .pipe(gulp.dest('dist'));
});
```

Sourcemap - gulp-sourcemaps

- sourcemaps 플러그인을 지원하는 플러그인 목록

HTML minification - gulp-htmlmin

- html 파일 크기 최소화 플러그인

```
var htmlmin = require('gulp-htmlmin');

gulp.task('minify', function() {
  return gulp.src('src/*.html')
    .pipe(htmlmin({collapseWhitespace: true}))
    .pipe(gulp.dest('dist'));
});
```

CSS minfication - gulp-cssnano

- CSS 용량 최소화 하는 플러그인

```
var cssnano = require('gulp-cssnano');

gulp.task('default', function() {
    return gulp.src('./main.css')
        .pipe(cssnano())
        .pipe(gulp.dest('./out'));
});

// Using Sourcemaps
var cssnano = require('gulp-cssnano');
var sourcemaps = require('gulp-sourcemaps');

gulp.task('default', function () {
    return gulp.src('main.css')
        .pipe(sourcemaps.init())
        .pipe(cssnano())
        .pipe(sourcemaps.write('.'))
        .pipe(gulp.dest('./out'));
});
```

CSS optimzation - gulp-uncss

- 빌드 시점에 사용하지 않는 css 들을 모두 제외

```
var uncss = require('gulp-uncss');

gulp.task('default', function () {
    return gulp.src('site.css')
        .pipe(uncss({
            html: ['index.html', 'posts/**/*.html', 'http://example.com']
        }))
        .pipe(gulp.dest('./out'));
});
```

Image minification - gulp-imagemin

- 이미지 파일의 품질을 떨어뜨리지 않는 선에서 용량을 최소화 한다.

```
var gulp = require('gulp');
var imagemin = require('gulp-imagemin'),
    pngquant = require('imagemin-pngquant');

gulp.task('default', function() {
    return gulp.src('src/images/*')
        .pipe(imagemin({
          progressive: true
        }))
        .pipe(gulp.dest('dist/images'));
});
```

Image optimization - gulp-image-optimzation

- imagemin 플러그인과 함께 사용

```
var gulp = require('gulp');
var imageop = require('gulp-image-optimization');

gulp.task('images', function(cb) {
    return gulp.src(['src/**/*.png','src/**/*.jpg','src/**/*.gif','src/**/*.jpeg']).pipe
        optimizationLevel: 5,
        progressive: true
    })).pipe(gulp.dest('public/images')).on('end', cb).on('error', cb);
});
```

File minification - gulp-filesize

- 파일 최적화를 진행 후 파일 사이즈를 확인할 수 있는 플러그인

```
var size = require('gulp-filesize');

gulp.src('./css/*.css')
    //all your gulp tasks
    .pipe(gulp.dest('./dist/'))
    .pipe(size()); // [gulp] Size example.css: 265.32 kB
```

Error hanlder - Pump

- Gulp 에서 일반적인 task 작업 구현시 .pipe() 를 사용하다가 발생한 에러에 대한 처리를 담당
- Gulp 에서 처리하는 기본적인 Exception 으로 로그를 찍는게 아니라, 정확히 어느 작업에서 오류가 났는지 명시

참고

- Highly Useful Gulp Plugins
- Scotch Gulp Tutorials
- Gulp for Beginners

끝