<http://programmingsummaries.tistory.com/356>

https://velopert.com/1344



[**[Node.js] 12.1 편: GULP – JavaScript 빌드 자동화툴 알아보기 + ES6 문법으로 사용해보기**](https://velopert.com/1344)

[2016년 5월 8일](https://velopert.com/1344) [velopert](https://velopert.com/author/velopert)[dev.log](https://velopert.com/category/dev-log) / [node.js](https://velopert.com/category/dev-log/tech-log/nodejs) / [tech.log](https://velopert.com/category/dev-log/tech-log)[4 Comments](https://velopert.com/1344#disqus_thread)

**[](https://velopert.com/node-js-tutorials)**

이번 강좌에서는 JavaScript 빌드 자동화 툴인 gulp.js 에 대하여 알아보고, ES6 문법으로 gulp.js 를 사용하는 방법을 알아보겠습니다.

**# 소개**

Node.js 환경에서 웹 어플리케이션을 만들다보면, 일일히 수작업으로 하기에 귀찮은 작업들이 존재합니다.. 예를들어서, \_\_\_\_.min.js, \_\_\_\_.min.css 이런 파일, 익숙하신가요? whitespace, newline 과 같이 없어도 지장이 되지 않는 문자들을 제거함으로서 페이지 렌더링 성능도 (비록 큰 차이는 아니지만) 를 늘리고 트래픽도 많이 아낄 수 있죠. jQuery 2.1.3 버전의 경우 uncompressed 와 minified 의 파일 사이즈가 159KB 차이가 난답니다 [1]. 하루 방문자가 1000명이라면, 155MB 의 트래픽을 아낄 수 있겠죠? 그만큼 중요한 file minification 작업을 js 와 css를 수정 할 때마다 수동으로 실행해야 한다면.. 귀찮겠죠? 다른 귀찮음의 예로는, Node.js 프로젝트를 작성하면서 .js 파일을 수정 할 때마다 서버를 재시작 해야하지 않았나요?

개발자들의 이런 귀차니즘을 해결하기 위한 도구가 바로 **gulp.js** 이랍니다. 위에 설명 된 것 외에도 많은 작업들을 다 자동으로 해주지요.

**# 설치하기**

**1. Gulp 전역(Global) 설치하기**

$ sudo npm install -g gulp

도중에  graceful-fs 와 lodash 에 관한 경고가 뜨면, 최신버전으로 설치해주세요.

sudo npm install -g graceful-fs lodash

/usr/local/lib

├── graceful-fs@4.1.3

└── lodash@4.11.2

**2. 프로젝트 폴더에서 npm init**

$ npm init

**3. gulp 와 gulp-util 를 devDependencies 로 모듈 설치**

(gulp-util 은 gulp에서 로그를 쉽게 기록 할 수 있게 해줍니다)

$ npm install -save-dev gulp gulp-util

**4. babel-core 와 babel-preset-es2015 를  devDependencies 로 모듈 설치**

$ npm install --save-dev babel-core babel-preset-es2015

위 모듈들은 gulp에서 ES6 를 사용 할 때 필요한 모듈들입니다.

**4. .babelrc 파일 생성**

{

"presets": ["es2015"]

}

스크립트를 변환해주는 모듈인 babel 의 설정입니다.  
ES6 문법을 사용하겠다는 의미입니다.

**5. gulpfile.babel.js 작성**

'use strict';

import gulp from 'gulp';

import gutil from 'gulp-util';

gulp.task('default', () => {

return gutil.log('Gulp is running');

});

**ES6 ALERT!**

import \_\_\_ from '\_\_\_' 는 var \_\_\_ = require('\_\_\_') 의 ES6 문법입니다.

() => { } 형태의 코드는 ES6 에 도입된 [“Arrow Function”](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Functions/%EC%95%A0%EB%A1%9C%EC%9A%B0_%ED%8E%91%EC%85%98) *[3]* 입니다. function(){ } 와 같은 의미이죠.

**gulpfile** 은 gulp 에서 어떤 작업들을 할 지 정의해줍니다.

* **LINE 1:**'use strict'; 는 JavaScript 코드의 안정성을 위하여 문법검사를 더 확실하게 하겠다는 의미입니다. 자세한 내용은 [“자바스크립트에서 strict mode를 사용해야 하는 이유”](http://blog.aliencube.org/ko/2014/01/02/reasons-behind-using-strict-mode-while-coding-javascript/) *[2]* 를 참고해주세요.

**6. gulp 실행**

$ gulp

[03:18:18] Requiring external module babel-register

[03:18:19] Using gulpfile ~/node\_tutorial/gulp-es6-webpack/gulpfile.babel.js

[03:18:19] Starting 'default'...

[03:18:19] Gulp is running

[03:18:19] Finished 'default' after 7.56 ms

기본 설정이 완료되었습니다!

**# 디렉토리 구조**

gulp-es6-webpack/

├── .babelrc

├── dist

├── node\_components

├── server

│ └── main.js

└── src

├── css

│ └── style.css

├── images

│ └── image.png

├── index.html

└── js

└── main.js

├── gulpfile.babel.js

├── index.js

├── package.json

gulpfile 을 추가적으로 작성하기 전에 저희 예제 프로젝트의 디렉토리 구조를 알아봅시다.

**src** 폴더 에는 Front-end 사이드에서 사용할 파일들이 있으며 gulp에서 minify 하여 dist 폴더에 변환된 파일들을 저장 할 것입니다.

**server** 폴더의 경우 server 사이드에서 사용 할 파일들이 있습니다. 이 프로젝트에선 server 부분에서도 ES6 를 쓸 것인데,  
이에 대해서는 다음 강좌에서 설명 할 예정입니다.

js 파일 및 css 파일들은 마음대로 작성하세요.

image 또한 원하는 이미지를 넣으세요.

**# GULP 한눈에 보기**

gulp 에는 4가지의 주요 API가 있습니다 *[4]*:

1. gulp.task
2. gulp.src
3. gulp.dest
4. gulp.watch

**gulp.task**(name *[, deps, fn]*) 는 gulp가 처리할 **task**, 즉 ‘**작업**‘ 을 정의합니다.

인수 **name** 은 string 형태로서 task의 이름을 지정하며, ***deps***와 ***fn*** 은 optional 인수로서, 생략되어도 되는 인수입니다.  
***deps***는 task name 의 배열 형태이며 이 인수가 전달 될 시, 이 배열 안에 있는 task들을 먼저 실행 한다음에, 함수형태로 전달되는 ***fn*** 을 실행합니다.

코드를 통해 봐볼까요?

gulp.task('hello', () => {

console.log('hello');

});

gulp.task('world', ['hello'], () => {

console.log('world');

});

이렇게 만든 task 는, 명령어 gulp **name** 을 통해 커맨드라인에서 특정 task를 실행 할 수 있습니다.

$ gulp world

[15:20:52] Requiring external module babel-register

[15:20:54] Using gulpfile ~/node\_tutorial/gulp-es6-webpack/gulpfile.babel.js

[15:20:54] Starting 'hello'...

hello

[15:20:54] Finished 'hello' after 268 μs

[15:20:54] Starting 'world'...

world

[15:20:54] Finished 'world' after 120 μs

gulp 명령어를 실행 할 때, name 을 명시하지 않으면 **default** task 가 실행됩니다.

**gulp.src**(globs*[, options]*) 는 어떤 파일을 읽을지 정합니다.

인수 **glob** 은 string 형태나 array 형태입니다. [node-glob syntax](https://github.com/isaacs/node-glob)*[5]* 를 사용하여 “\*\*/\*.js” 이런식으로 여러 파일을 한꺼번에 지정 할 수 있습니다.  
**options**는 Object 형태이며 node-glob에 전달 할 옵션입니다. 자세한 내용은 [GULP API](https://github.com/gulpjs/gulp/blob/master/docs/API.md#gulpsrcglobs-options)*[4]*를 확인해주세요.

이 함수가 리턴한 객체에서는 .pipe 를 통하여 다른 플러그인을 사용해 변환 할 수 있습니다.

이에 대한 예제는 잠시 후 플러그인을 설치하고 gulpfile 을 작성 할 때 알아보겠습니다.

**gulp.dest**(path[, options]) 는 어디에 저장할지 정합니다.

**path** 는 디렉토리를 입력하며,

**options**는 객체로서 { **cwd**: \_\_\_\_, mode: \_\_\_\_ } 형태입니다.  
**cwd** 는 현재 디렉토리 위치로서 .path가 /build/ 이런식으로 상대적일때 현재 디렉토리를 따로 설정하고 싶을 때 사용하며,  
**mode** 는 파일권한 (기본 : “0777”) 입니다.

이에 대한 예제 또한 gulpfile을 작성 할 때 알아보겠습니다.

**gulp.watch**(glob[, opts], tasks/cb) 는 전달된 **glob**에 해당하는 파일들을 주시하고있다가, 변동이 있을 시 **tasks**를 실행합니다.

인수 **tasks**는 task **name**의 배열형태입니다. 배열 형태가 아닐 땐 **event**를 파라미터로 가지고있는 콜백함수 **cb**를 작성합니다.  
**opts**는 gulp에서 사용하는 라이브러리인 [gaze](https://github.com/shama/gaze)*[6]* 에 전달 할 옵션입니다.

매뉴얼에 적혀있는 예제를 한번 훑어봅시다:

var watcher = gulp.watch('js/\*\*/\*.js', ['uglify','reload']);

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

console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');

});

// OR

gulp.watch('js/\*\*/\*.js', function(event) {

console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');

});

**# gulpfile 작성하기**

gulp에 대한 간단한 설명을 봤으니, 이제 gulpfile을 작성해볼까요?

먼저, 우리가 뭘 하고 싶은지 적어봅시다.

* minify javascript
* minify css
* minify html
* compress image

gulp 자체에서는 위 기능들을 지원하지 않습니다. 단! gulp 플러그인들이 위 역할들을 대신해주지요.

**# 플러그인 설치하기**

gulp 플러그인들의 갯수는 2016년 5월 기준으로 2375개나 있습니다. 플러그인 검색은 [Gulpjs 홈페이지](http://gulpjs.com/plugins/)*[7]* 에서 하실 수 있습니다.

저희가 사용 할 플러그인은 다음과 같습니다:

* [gulp-uglify](https://www.npmjs.com/package/gulp-uglify/)
* [gulp-clean-css](https://www.npmjs.com/package/gulp-clean-css/)
* [gulp-htmlmin](https://www.npmjs.com/package/gulp-htmlmin/)
* [gulp-imagemin](https://www.npmjs.com/package/gulp-imagemin/)
* [del](https://www.npmjs.com/package/del)

위 링크를 클릭하시면 플러그인의 사용법도 나옵니다.

del 모듈은 gulp 플러그인은 아닙니다. gulp 플러그인으로 제작된 모듈이 아니더라도 gulpfile 내에서 사용 할 수는 있답니다.  
이 모듈은 특정 디렉토리를 삭제해주는 플러그인입니다. 동기식으로 삭제 할 수 있는 기능을 가지고 있죠.

gulp 작업이 실행 될 때 마다 기존 dist 디렉토리에 있는 파일들을 삭제해줘야 하기 때문에 이 플러그인을 사용합니다.

npm 을 통하여 설치해봅시다.

$ npm install --save-dev gulp-uglify gulp-clean-css gulp-htmlmin gulp-imagemin del

설치가 끝났다면 gulpfile 상단에 위 플러그인들을 import 해주세요.

import uglify from 'gulp-uglify';

import cleanCSS from 'gulp-clean-css';

import htmlmin from 'gulp-htmlmin';

import imagemin from 'gulp-imagemin';

import del from 'del';

**# 디렉토리 정의**

먼저, 소스/빌드 디렉토리를 담은 객체를 만들어봅시다.  
이 과정은 필수가 아니지만, 이렇게 하면 코드가 간결해지고 나중에 수정하기도 편하답니다.

const DIR = {

SRC: 'src',

DEST: 'dist'

};

const SRC = {

JS: DIR.SRC + '/js/\*.js',

CSS: DIR.SRC + '/css/\*.css',

HTML: DIR.SRC + '/\*.html',

IMAGES: DIR.SRC + '/images/\*'

};

const DEST = {

JS: DIR.DEST + '/js',

CSS: DIR.DEST + '/css',

HTML: DIR.DEST + '/',

IMAGES: DIR.DEST + '/images'

};

**ES6 ALERT!**[const](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Statements/const) 는 ES6 문법에 도입된 읽기전용 값인 상수를 선언 할 때 사용됩니다.

**# TASK 작성하기**

**# minify javascript**

gulp.task('js', () => {

return gulp.src(SRC.JS)

.pipe(uglify())

.pipe(gulp.dest(DEST.JS));

});

코드를 저장하고 실행해보세요.

$ gulp js

[18:48:43] Requiring external module babel-register

[18:48:43] Working directory changed to ~/node\_tutorial/gulp-es6-webpack

[18:48:43] Using gulpfile ~/node\_tutorial/gulp-es6-webpack/gulpfile.babel.js

[18:48:43] Starting 'js'...

[18:48:43] Finished 'js' after 74 ms

문제없이 실행됐나요? dist/js 폴더에 파일이 저장되었는지도 확인해보세요. 그렇다면 다음 단계로 넘어갑시다.

**# minify css**

gulp.task('css', () => {

return gulp.src(SRC.CSS)

.pipe(cleanCSS({compatibility: 'ie8'}))

.pipe(gulp.dest(DEST.CSS));

});

**# minify html**

gulp.task('html', () => {

return gulp.src(SRC.HTML)

.pipe(htmlmin({collapseWhitespace: true}))

.pipe(gulp.dest(DEST.HTML))

});

**# compress images**

gulp.task('images', () => {

return gulp.src(SRC.IMAGES)

.pipe(imagemin())

.pipe(gulp.dest(DEST.IMAGES));

});

**# clean**

gulp.task('clean', () => {

return del.sync([DIR.DEST]);

});

**# default**

자, 이제 기본 gulp task 를 정의 할 차례입니다. 기본 task 에서는 위에 만든 여러 task 들을 실행하도록 설정하겠습니다.

gulp.task('default', ['clean', 'js', 'css', 'html', 'images'], () => {

gutil.log('Gulp is running');

});

**gulp** 명령어를 입력해서 테스트 해봅시다.

$ gulp

[02:49:37] Requiring external module babel-register

[02:49:38] Using gulpfile ~/node\_tutorial/gulp-es6-webpack/gulpfile.babel.js

[02:49:38] Starting 'clean'...

[02:49:38] Finished 'clean' after 22 ms

[02:49:38] Starting 'js'...

[02:49:38] Starting 'css'...

[02:49:38] Starting 'html'...

[02:49:38] Starting 'images'...

[02:49:38] Finished 'js' after 85 ms

[02:49:38] Finished 'html' after 220 ms

[02:49:38] Finished 'css' after 238 ms

[02:49:40] gulp-imagemin: Minified 1 image (saved 17.06 kB - 13.2%)

[02:49:41] Finished 'images' after 2.26 s

[02:49:41] Starting 'default'...

[02:49:41] Gulp is running

[02:49:41] Finished 'default' after 444 μs

오류 없이 실행 됐나요?

**# WATCH 작성하기**

watch는 특정 디렉토리 및 파일들을 감시하고 있다가 변동이 감지 될 시, 지정한 task 를 실행시키는 기능입니다.

gulp.task('watch', () => {

gulp.watch(SRC.JS, ['js']);

gulp.watch(SRC.CSS, ['css']);

gulp.watch(SRC.HTML, ['html']);

gulp.watch(SRC.IMAGES, ['images']);

});

watch 를 작성하는건 위와 같이 간단합니다. 첫번째 인수로 전달된 값에 해당하는 파일들을 감시하고 있다가, 두번째 인수로 전달된 task 를 실행합니다.

어떤 파일이 변경되었는지 기록하고싶다면, 코드를 다음과 같이 수정하세요.

gulp.task('watch', () => {

let watcher = {

js: gulp.watch(SRC.JS, ['js']),

css: gulp.watch(SRC.CSS, ['css']),

html: gulp.watch(SRC.HTML, ['html']),

images: gulp.watch(SRC.IMAGES, ['images'])

};

let notify = (event) => {

gutil.log('File', gutil.colors.yellow(event.path), 'was', gutil.colors.magenta(event.type));

};

for(let key in watcher) {

watcher[key].on('change', notify);

}

});

그리고, ‘watch’ 를 **default** task 부분의 deps 배열에 넣어주세요.

gulp.task('default', ['clean', 'js', 'css', 'html', 'images', 'watch'], () => {

gutil.log('Gulp is running');

});

자, 테스트를 해봐야겠죠? 새로운 screen 을 열어서 gulp 를 실행 시킨 다음에 파일을 수정해보세요.

$ screen -S gulp

#####################################################################

$ gulp

[03:25:49] Requiring external module babel-register

[03:25:50] Using gulpfile ~/node\_tutorial/gulp-es6-webpack/gulpfile.babel.js

[03:25:50] Starting 'clean'...

[03:25:50] Finished 'clean' after 11 ms

[03:25:50] Starting 'js'...

[03:25:50] Starting 'css'...

[03:25:50] Starting 'html'...

[03:25:50] Starting 'images'...

[03:25:50] Starting 'watch'...

[03:25:50] Finished 'watch' after 20 ms

[03:25:50] Finished 'html' after 154 ms

[03:25:50] Finished 'js' after 178 ms

[03:25:50] Finished 'css' after 165 ms

[03:25:52] gulp-imagemin: Minified 1 image (saved 17.06 kB - 13.2%)

[03:25:52] Finished 'images' after 2 s

[03:25:52] Starting 'default'...

[03:25:52] Gulp is running

[03:25:52] Finished 'default' after 308 μs

#####################################################################

# CTRL+A+D, edit files ...

#####################################################################

$ screen -r gulp

#####################################################################

[03:28:02] File /home/vlpt/node\_tutorial/gulp-es6-webpack/src/index.html was changed

[03:28:02] Starting 'html'...

[03:28:02] Finished 'html' after 35 ms

[03:28:09] File /home/vlpt/node\_tutorial/gulp-es6-webpack/src/js/main.js was changed

[03:28:09] Starting 'js'...

[03:28:09] Finished 'js' after 11 ms

[03:28:21] File /home/vlpt/node\_tutorial/gulp-es6-webpack/src/css/style.css was changed

[03:28:21] Starting 'css'...

[03:28:21] Finished 'css' after 29 ms

성공!



**# 마치면서..**

이번 포스트에서는 gulp 에 대한 기본적인 사용법을 알아봤습니다.  
사용된 코드는 [GitHub](https://github.com/velopert/gulp-es6-webpack/tree/tut12-1) 에서 열람 가능합니다.  
파일을 minify 하는 것 외에도 gulp 는 정말 많은 것 들을 할 수 있답니다.

아직 gulp 에 대하여 공부 할 것들이 많이 남아있답니다.

앞으로 다룰 것들은 다음과 같습니다:

* 서버 사이드 에서 ES6 사용
* express 로 웹서버 열고, 서버파일이 수정 될 떄마다 서버 재시작
* 프론트엔드 사이드의 파일들이 변경 될 시, 브라우저 자동으로 새로고침
* webpack 을 사용하여 프론트엔드 사이드에서 ES6 을 사용하고 import 도 사용하기

기대되지 않나요? 🙂 다음 강좌를 기다려주세요 ~

**References**

1. [“jQuery file size”](https://mathiasbynens.be/demo/jquery-size). *Mathias.*
2. [“자바스크립트에서 strict mode를 사용해야 하는 이유”](http://blog.aliencube.org/ko/2014/01/02/reasons-behind-using-strict-mode-while-coding-javascript/). *Aliencube Community.*
3. [“Arrow Function”](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Functions/%EC%95%A0%EB%A1%9C%EC%9A%B0_%ED%8E%91%EC%85%98). *Mozilla*
4. [“Gulp API”](https://github.com/gulpjs/gulp/blob/master/docs/API.md). *GitHub.*
5. [Glob”](https://github.com/isaacs/node-glob). GitHub.
6. [“Gaze”](https://github.com/shama/gaze). GitHub.
7. [“Gulp.js plugin registry”.](http://gulpjs.com/plugins/) Gulp.js.

**[](https://velopert.com/node-js-tutorials)**