Gulp Basics

작성자 : 배영
소속팀 / 상위부서 : UI개발1팀 / UIT개발실
대외비

목차

1.
Gulp 시작하기

1.1 Gulp 소개

- Gulp?
  - 정형화된 작업 프로세스를 task Script로 작성하여 자동으로 실행시켜주는 툴
    - 단일 업무를 수행할 수도 있고, 엮어서 하나의 pipeline으로도 실행 가능
    - 각 task에 다른 task를 dependency 주입 가능
  - 사용 예
    - Sass를 CSS로, CoffeeScript를 javaScript로 컴파일
    - 자바스크립트 파일을 하나로 단일화 (concatenate)
    - 자바스크립트 파일을 한줄로 압축 (minify)
    - task 감시 (watch)
    - task 파일 구조화 (scaffold)

- 개발자가 주로 해야 할 일은 코드에 집중하는 것!
- gulp는 개발자가 코드에 집중할 수 있도록 다른 부가적인 업무를 자동으로 수행해주는 역할을 한다.

1.2 gulpfile.js

- gulpfile.js 프로젝트 루트에 위치
- gulp가 실행해야 할 task가 기록되어 있음
- gulp 커맨드 실행 시 '**default task**'가 실행됨

1.3 gulp 설치하기

1.gulp는 Node.js를 바탕으로 만들어 졌으므로 Node와 NPM이 먼저 설치되어 있어야 함

2.노드 프로젝트 생성

1) 해당 프로젝트에서 npm init 실행하여 노드 프로젝트 생성

2) npm install gulp --save-dev 실행하여 devDependencies에 gulp 추가

①gulp 사용은 실서비스 서버단에 필요한 것이 아니라 개발에 필요하므로 --save-dev 옵션을 준다.

- package.json 파일을 열어보면 devDependencies에 gulp가 추가된 것을 볼 수 있다.

②혹은 --global(-g) 옵션을 주어서 시스템 전체에서 사용가능하도록 설치 가능

③global과 local 설치에 관해서: https://nodejs.org/en/blog/npm/npm-1-0-global-vs-local-installation/

1.4 첫번째 gulp task 작성하기

- gulp 모듈을 require해서 불러오기
- task 작성

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

gulp.task("hello", function() {
    console.log("Hello!");
});
```

- 실행 화면: gulp hello

```
$ gulp hello
[18:14:03] Using gulpfile ~\Downloads\treehouse-gulp-basics\gulpfile.js
[18:14:03] Starting 'hello'...
Hello!
[18:14:03] Finished 'hello' after 86 µs
```

- Default task 추가하여 gulp 커맨드만으로 task 실행되게 하기

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

```
MINGW64:/c/Users/NAVER.AD00030654/Downloads/treehouse-gulp-basics
NAVER@AD00030654 MINGW64 ~/Downloads/treehouse-gulp-basics
$ gulp
[18:15:33] Using gulpfile ~\Downloads\treehouse-gulp-basics\gulpfile.js
[18:15:33] Starting 'hello'...
Hello!
[18:15:33] Finished 'hello' after 85 µs
[18:15:33] Starting 'default'...
[18:15:33] Finished 'default' after 6.53 µs
```

2.

자바스크립트에 gulp 적용해보기

Gulp-concat, gulp-uglify, gulp-rename 플러그인

2.1 Gulp 플러그인

- 노드 패키지 형태의 플러그인을 받아서 task 작성에 이용 할 수 있음
- 거의 필요한 모든 업무에 관한 플러그인이 이미 작성되어 있으므로 찾아보면 됨
  - http://gulpjs.com/plugins/

2.2 gulp-concat 플러그인

- 필요성: 배포시에 작업한 모든 스크립트 파일을 하나로 합쳐서 배포하면 파일 전송 시간이 줄어든다.
- 설치: npm install gulp-concat --save-dev
- uire 한 후, task 작성

```
gulp.task('concatScripts', function() {
    gulp.src(['js/jquery.js',
              'js/sticky/jquery.sticky.js',
              'js/main.js'])
    .pipe(concat("app.js"))
    .pipe(gulp.dest('js'));
});
```

- js 파일 나열 순서에 신경을 써서 작성
- src 메소드는 stream 형태로 파일을 메모리에 올림
- concat 메소드를 사용하여 인자로 결과물 파일의 이름 입력
- gulp.dest 메소드에 인자로 파일을 저장할 디렉터리 입력

2.3 gulp-uglify 플러그인

- 필요성: 파일을 한줄로 압축하여 파일 용량 줄임
- 설치: npm install gulp-uglify --save-dev

```
gulp.task('minifyScripts', function() {
    gulp.src('js/app.js')
        .pipe(uglify())
        .pipe(gulp.dest('js'));
});
```

  - gulpfile.js에서 gulp-uglify 모듈을 require 한 후, task 작성

- 문제점: app.js에 uglify한 파일이 덮어써지기 때문에 이 파일을 따로 저장하고 싶음!

2.4 gulp-rename 플러그인

- 파일 덮어쓰기 방지
- 설치: npm install gulp-rename --save-dev

```
gulp.task('minifyScripts', function() {
    gulp.src('js/app.js')
        .pipe(uglify())
        .pipe(rename('app.min.js'))
        .pipe(gulp.dest('js'));
});
```

  - rename 메소드의 인자로 바꿀 이름을 입력

- gulp 모듈 require시 변수 이름을 잘 정하는 것이 중요!

3.

CSS에 gulp 적용해보기

Gulp-sass, gulp-sourcemaps 플러그인

3.1 SaSS를 CSS로 컴파일 해보기

- Sass: 브라우저가 파일을 읽기 전에 CSS파일로 컴파일 되어야 함
  - 컴파일 별로 어렵지 않음. 그러나, 시간 낭비!
- Gulp: 자동 컴파일을 도와 준다.
  - 수정시: Sass에서 컴파일된 CSS파일을 직접 수정하는 것은 금물
    - 왜? sass에서 다시 수정 후 컴파일하면 기존 CSS파일에 덮어쓰기가 되므로 변경사항이 날라간다
    - 따라서, gulp에서 자동 컴파일을 위한 task를 작성한다.

3.1 SaSS를 CSS로 컴파일 해보기

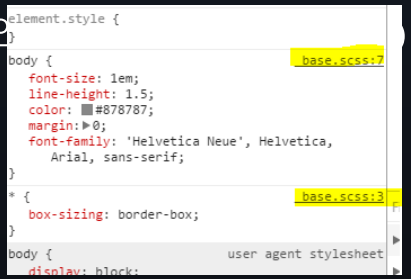
- Gulp를 사용해서 Sass 자동 컴파일 task 작성해보기
  - npm을 사용하여 gulp-sass 플러그인 설치: npm install gulp-sass --save-dev
  - gulpfile.js에서 모듈 require한 뒤에 Sass 컴파일을 위한 task 코드 작성

```
gulp.task('compileSass', function () {
   gulp.src('scss/application.scss')
        .pipe(sass()) // Sass를 CSS로 컴파일
        .pipe(gulp.dest('css')) // 파일 저장될 폴더 설정
});
```

  - 왜 src파일에 application.scss 파일만 첨부하나요?
    - application.scss 파일에서 base, component, layout에 해당하는 scss파일을 import하므로 이 파일만 사용하면 자동으로 엮여서 컴파일이 된다.

3.2 SaSS에 Source Map 추가해보기

- 컴파일 된 CSS에 수정을 하고 싶을 때, SaSS 파일이 많다면 어디서 고쳐야 할 지 막막할 수도 있음
  - Source Map을 사용한다!
  - Source Map이 존재할 때: 웹 브라우저 개발자 도구에서 sass파일을 바로 확인
  - Soruce Map이 없을 때: 직접 찾아가야 함. 귀찮음.



- Source Map 추가해보기
  - npm을 사용하여 gulp-sourcemaps 플러그인 설치:

    npm install gulp-sourcemaps --save-dev
  - p 생성을 위한 task 코드 작성

```
gulp.task('compileSass', function() {
  gulp.src("scss/application.scss")
      // sass 컴파일 하기 전에 source map 초기화 실행
      .pipe(maps.init())
      .pipe(sass())
      // source map 파일 생성. 경로 지정해줌
      // 경로: gulp.dest 디렉터리의 상대 경로
      .pipe(maps.write('./'))
      .pipe(gulp.dest('css'));
});
```

3.3 자바스크립트에 Source Map 추가해보기

- 자바스크립트 Source Map이 없을 때
  - 스크립트 파일을 하나로 묶은 후 (gulp-concat 모듈 등을 사용) 어디서 에러가 났는지 찾으려고 할 때 js 파일이 많으면 많을수록 에러 찾기가 힘듦
- source map을 사용하면?
  - 사용방법은 css파일에 source map 적용할 때와 동일
  - 스크립트 파일을 하나로 합쳐놓아도 개발자 도구 등에서 합치기 전의 소스 코드 위치로 나타나기 때문에 문제 되는 부분을 찾기 쉬움

4.

Pipeline 보충

4.1 Gulp task에 dependency 추가하기

- Task runner 장점: 여러 task를 하나로 엮을 수 있음
- 하나 이상의 task를 한번에 커맨드 라인에서 실행하려면?
  - 하나씩 입력하기에는 일이 많음. 프로젝트가 커질수록 힘듦.

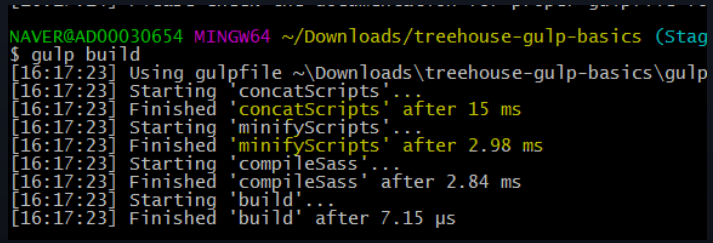
- gulp.task("build")

```
gulp.task("build", ['concatScripts', 'minifyScripts', 'compileSass']);
```

  - 문제점?
    - gulp는 작업 속도를 줄이기 위해 모든 task를 동시다발적으로 실행함
    - task의 실행 순서가 꼬여서 작업 결과물이 예상과 다를 수 있음

4.1 Gulp task에 dependency 추가하기

- 시나리오
  - minifyScripts(스크립트 압축)는 concatScripts(스크립트를 파일 하나로 붙이기)보다 일찍 task가 끝나는데, minify는 concat이 만들어내는 app.js를 필요로 하므로, 결과적으로는 완전한 압축을 할 수 없다.



  - 따라서, 이를 해결하기 위해 concat 작업 후에 minify가 실행되도록 해야 한다!

4.1 Gulp task에 dependency 추가하기

- **이를 해결하기 위해 Gulp가 제공하는 기능**
  - minifyScripts의 dependency로 concatScripts를 추가한다.

```
gulp.task("minifyScripts", ["concatScripts"], function() {
    return gulp.src("js/app.js")
        .pipe(uglify())
        .pipe(rename('app.min.js'))
        .pipe(gulp.dest('js'));
});
```

  - concatScripts의 콜백 함수에 **return 문**을 추가한다.
    - return문이 존재하지 않으면 다른 task들이 concatScripts가 끝났는지 아닌지 알 수가 없다.

  - build task에서 concatScripts를 삭제한다. (dependency로 추가되었으므로)

```
gulp.task("build", ['minifyScripts', 'compileSass']);
```

4.1 Gulp task에 dependency 추가하기

- **결과**
  - concatScripts task가 minifyScripts 시작 전에 끝남

```
[16:28:50] Using gulpfile ~\Downloads\treehouse-gulp-basics\gulpfile.js
[16:28:50] Starting 'concatScripts'...
[16:28:50] Starting 'compileSass'...
[16:28:50] Finished 'compileSass' after 84 ms
[16:28:50] Finished 'concatScripts' after 103 ms
[16:28:50] Starting 'minifyScripts'...
[16:28:51] Finished 'minifyScripts' after 1.51 s
[16:28:51] Starting 'build'...
[16:28:51] Finished 'build' after 14 µs
```

- **default task**에 작성한 build task를 추가해주면
  커맨드 창에서 **gulp만 입력** 시 모든 task가 자동으로 수행된다.

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

4.2 gulp 내장 메소드 'watch'

- **watch 메소드**란? 파일이 바뀌는 것을 지켜보다가 변경사항이 생기면 콜백 함수로 작성한 task를 실행시킴
- 작성하기

1. watch 메소드에 Sass partials(Sass파일을 모듈화할 때 사용하는 파일들)를 첫 번째 파라미터로 넘김
   - 배열로 파일 목록을 넘김: 파일 수가 많을 수록 힘듦

```
gulp.task("watchSass", function () {
   // watch할 파일을 parameter로 넘김
   gulp.watch(['scss/application.scss', 'scss/base/_base.scss', ...])
});
```

   - globbing pattern: 패턴을 작성하여 그 패턴에 해당하는 이름을 가진 파일만 파라미터로 넘김

```
gulp.task("watchSass", function () {
   // watch할 파일을 parameter로 넘김
   gulp.watch(['scss/**/*.scss'])
});
```

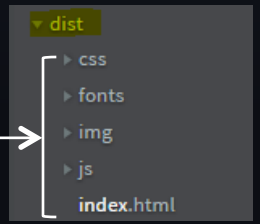
2. 두번째 파라미터는 해당 task 이름
   - task 이름 작성 시, string이 아니라 배열로 작성해야 함!!

```
gulp.task("watchSass", function () {
   // watch할 파일을 parameter로 넘김
   gulp.watch('scss/**/*.scss', ['compileSass'])
});
```

4.3 개발 & 배포 Pipeline

- Gulp 사용 환경
  - 앱 배포 시
  - 앱 개발 시
- 배포 Pipeline
  - pipeline 마지막 단계: **dist 폴더** 생성해서 작업물을 넣기
    - build task의 마지막 parameter로 익명 함수 생성
      - src의 마지막 파라미터로 **base 옵션** 주기: gulp한테 소스로 제공되는 모든 파일의 *디렉터리 구조를 그대로 보존*하라고 알려주는 역할

```
gulp.task("build", ['minifyScripts', 'compileSass'], function() {
    return gulp.src(["css/application.css", "js/app.min.js", "index.html",
                     "img/**", "fonts/**"], { base: './' })
                .pipe(gulp.dest('dist'));
});
```



4.4 조금 더 쓸만한 Pipeline 만들기

- watch 메소드 추가하기
  - 한 줄로 함수를 작성할 수도 있지만, 합쳐놓을 시 실행될 필요가 없는 task까지 실행하기 때문에 그냥 *구분해서 따로따로 작성*하는 편이 효율적인 task가 된다.

```
gulp.task('watchFiles', function() {
  gulp.watch('scss/**/*.scss', ['compileSass']);
  gulp.watch('js/main.js', ['concatScripts']);
})
```

자료 출처

- 본 ppt는 아래 링크의 강의 내용을 요약해 놓은 것입니다.
  - https://teamtreehouse.com/library/gulp-basics
- 기타 참고 사이트
  - https://css-tricks.com/gulp-for-beginners/

-

End of Document

-

Thank You.

-