

박창우, @pismute

GULP로 정적 페이지 생성하기

Gulp

Automate and enhance your workflow

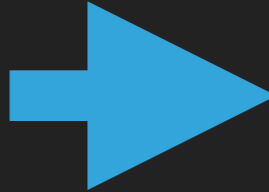
markdown



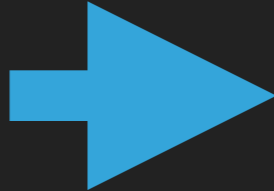




Build



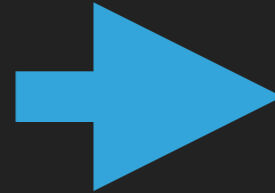


Build

>>>>>
livereload





Build



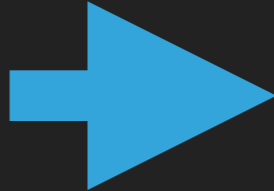
livereload



Deploy





Build

>>>>>
livereload

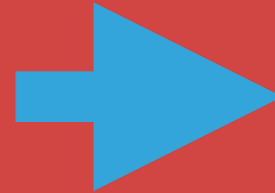


 Deploy





Build



livereload

Gulp



Deploy



```
gulp.task('build', ...);
```

```
gulp.task('handlebars', ...);
```

```
gulp.task('markdown', ...);
```

```
gulp.task('asciidoc', ...);
```

```
gulp.task('livereload', ...);
```

```
gulp.task('deploy', ...);
```

Fork me on GitHub

StaticGen

Top Open-Source Static Site Generators

SHARE

All languages

Sorted by stars

About StaticGen The Rules

Jekyll

jekyllrb.com

★	📄	👤
22158	4687	168
0	0	0

A simple, blog-aware, static site generator.

GitBook

www.gitbook.com/

★	📄	👤
10220	1055	307
0	0	0

A modern publishing toolchain. Simply taking you from ideas to finished, polished books.

Octopress

octopress.org

★	📄	👤
9202	3157	272
0	0	0

A blogging framework for hackers based on Jekyll.

Hexo

hexo.io/

★	📄	👤
7129	1182	187
0	0	0

Hexo is a fast, simple and powerful blog framework



굳이! 정적 페이지를 직접 만든 이유는?

- ▶ 결국 HTML은 **프론트엔드, 자동화**니까 NodeJS로...
- ▶ 기능을 Task 단위로 쪼개서 조립 해보자
 - ▶ 그리고, 어쩌면 재사용

발표 내용.

- ▶ Gulp를 어떤 느낌으로 사용하면 좋을지?
- ▶ 직접 정적 페이지를 생성하면서 고민했던 것들.



GULP

Task

```
gulp.task('myTask', function(){  
  return gulp.src(['**/*'])  
    .pipe(gulp.dest('dest'));  
});
```

gulp-load-plugins 모듈

```
// package.json에 정의한 모듈을
// 편리하게 쓸 수 있도록 도와준다.
var $ = require('gulp-load-plugins');

gulp.task('myTask', function(){
  return gulp.src(['**/*'])
    .pipe($.using()) // 'gulp-using' 모듈을 lazy하게 로딩
    .pipe($.size()) // 'gulp-size' 모듈을 lazy하게 로딩
    .pipe(gulp.dest(global.DEST));
});
```

Task 순서

```
gulp.task('serial2', ['serial1'], function() {  
    ...  
});
```

Task 순서: run-sequence

```
var seq = require('run-sequence');

gulp.task('task', function() {
  seq('serial1',
    ['parallel121', 'parallel122'],
    'serial2');
});
```


Task 끝내기: callback

```
var seq = require('run-sequence');

gulp.task('task', function(done) {
  seq('serial1',
    ['parallel1', 'parallel2'],
    'serial2',
    done);
});
```

Task 끝내기: return stream

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

Task 끝내기: return promise

```
var Q = require('q');

gulp.task('somename', function() {
  var deferred = Q.defer();

  // do async stuff
  setTimeout(function() {
    deferred.resolve();
  }, 1);

  return deferred.promise;
});
```

정적 페이지 만들기

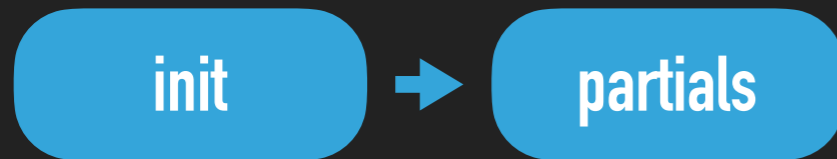
Directory Layout

```

.
├── app/
│   ├── layouts/
│   │   ├── **/.model.json // document model
│   │   └── **/*.hbs // document decorators
│   └── partials/
│       └── **/*.hbs // document components
├── docs/
│   ├── **/.model.json // document model
│   ├── **/*.html.md // docs, markdown to html
│   ├── **/*.html.asc // docs, asciidoc to html
│   ├── **/*.html.hbs // docs, handlebars to html
│   └── **/* // copy to dist/
├── root/
│   ├── **/.model.json // document model
│   ├── **/*.xml.hbs // template, handlebars to xml
│   ├── **/*.html.hbs // template, handlebars to html
│   └── **/* // copy to dist/
└── dist/ //target directory
  
```


Init

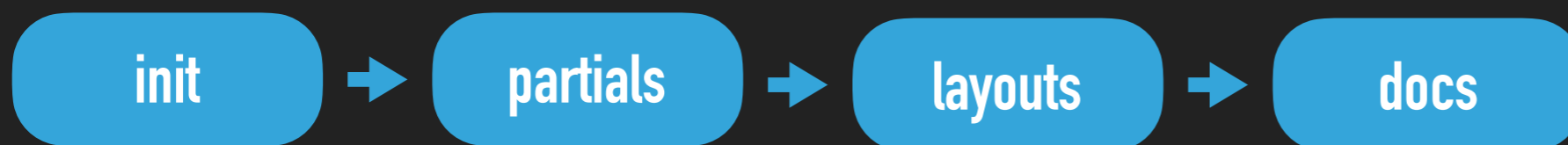
```
gulp.task('init', function(done) {  
  .....  
  
  global.DOCS = 'app/docs/**/*.{md,hbs,asc}'  
  global.TEMPLATES = ['app/root/**/*.hbs']  
  global.PARTIALS = 'app/partials/**/*.hbs'  
  global.LAYOUTS = 'app/layouts/**/*.hbs'  
  
  .....  
  done();  
});
```



```
gulp.task('partials', function(){  
  return gulp.src(global.PARTIALS)  
    .pipe(.....);  
});
```



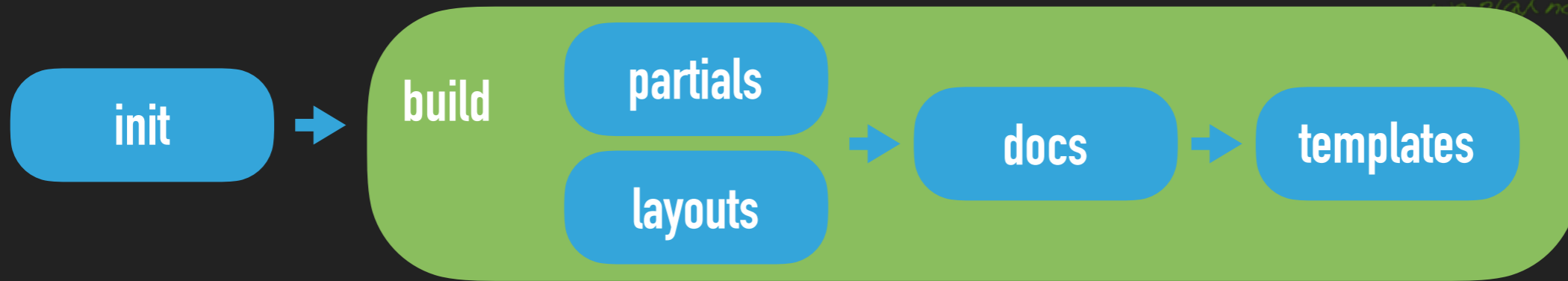
```
gulp.task('layouts', function(){  
  return gulp.src(global.LAYOUTS)  
    .pipe(.....);  
});
```



```
gulp.task('docs', function(){  
  return gulp.src(global.DOCS)  
    .pipe(layout/partial 적용)  
    .pipe(.....);  
});
```

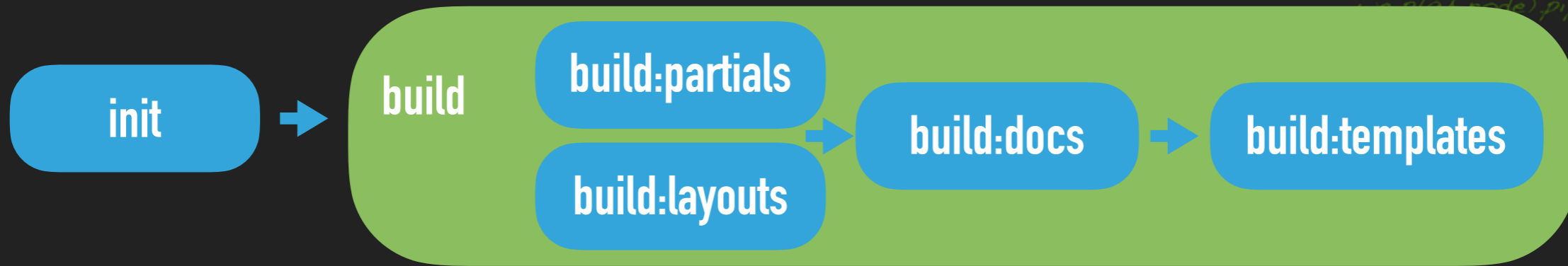


```
gulp.task('templates', function(){  
  return gulp.src(global.TEMPLATES)  
    .pipe(layout/partial 적용)  
    .pipe(.....);  
});
```



```
var seq = require('run-sequence');

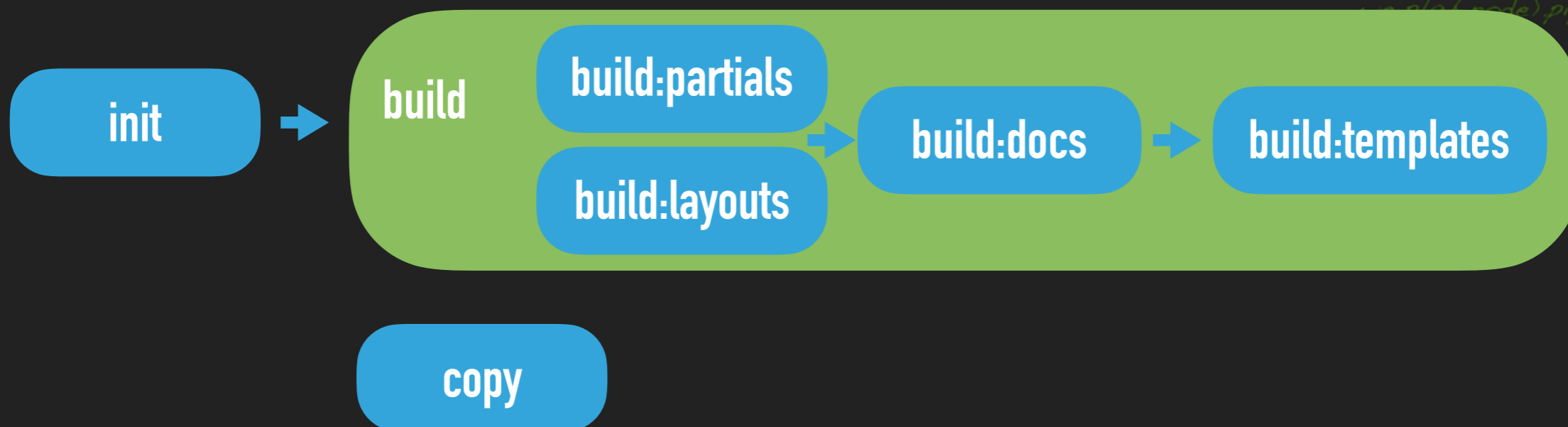
gulp.task('build', function(done) {
  seq(['partials', 'layouts',
    'docs',
    'templates',
    done]);
});
```



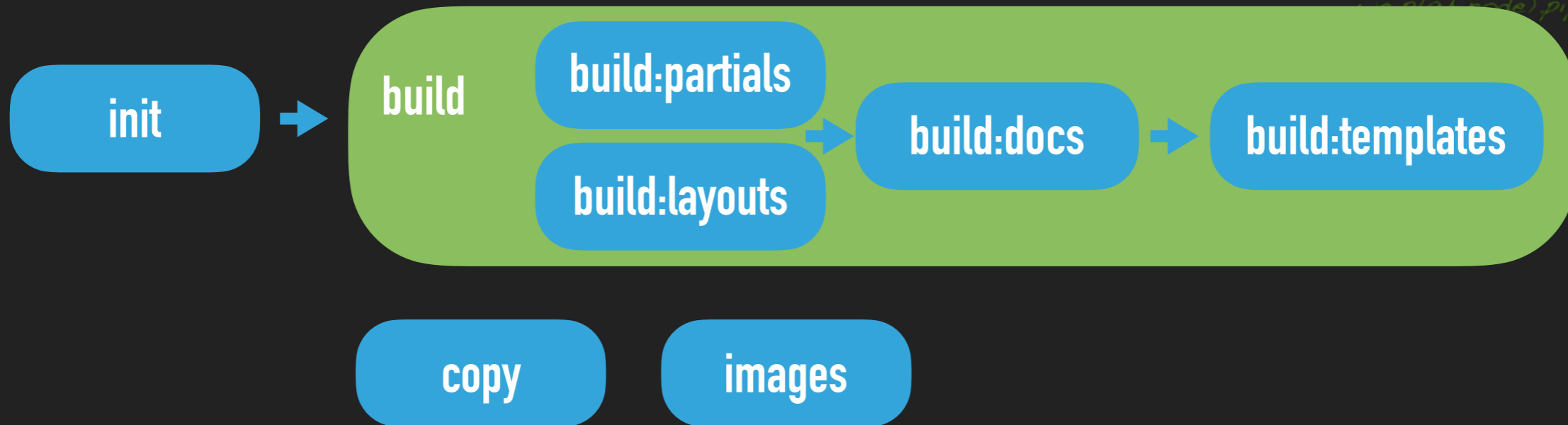
```
var seq = require('run-sequence');

gulp.task('build', function(done) {
  seq(['build:partials', 'build:layouts'],
    'build:docs',
    'build:templates',
    done);
});
```

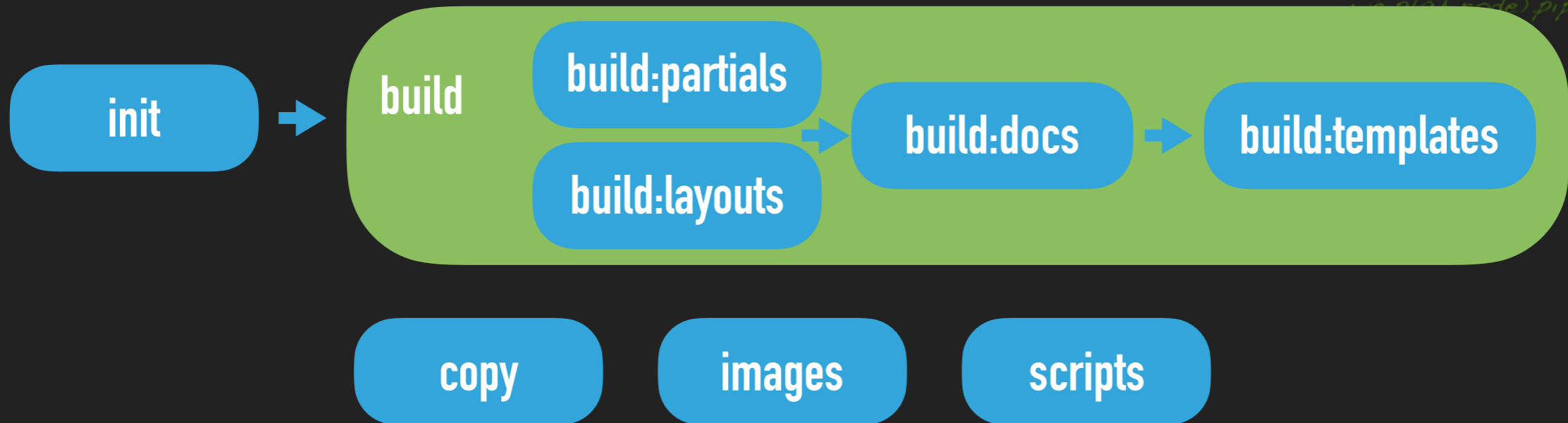
\$ gulp build:docs



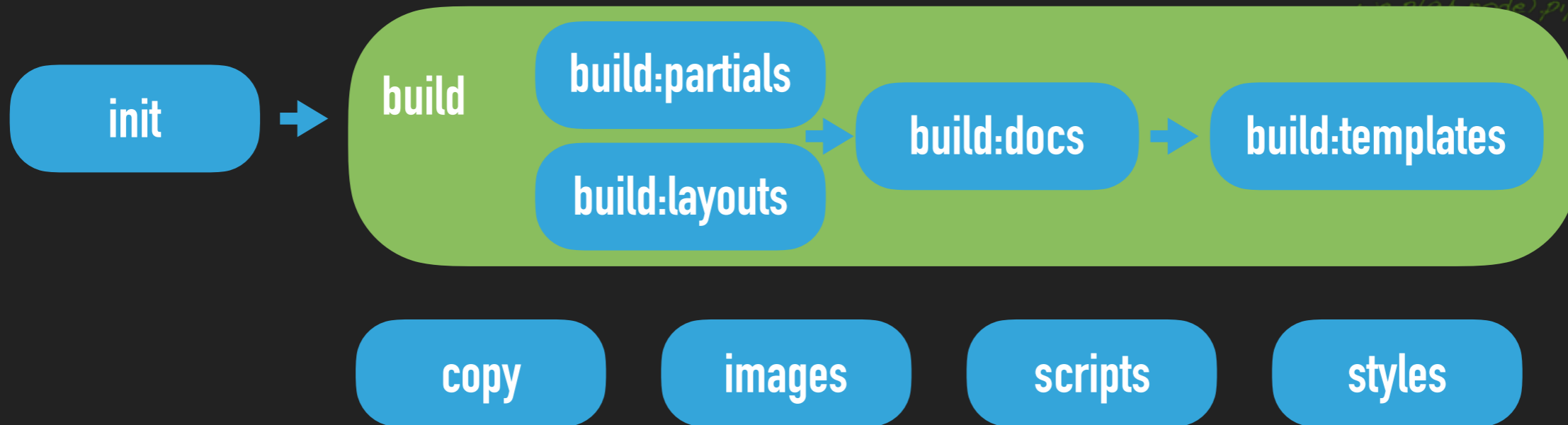
- ▶ 파일을 있는 그대로 복사



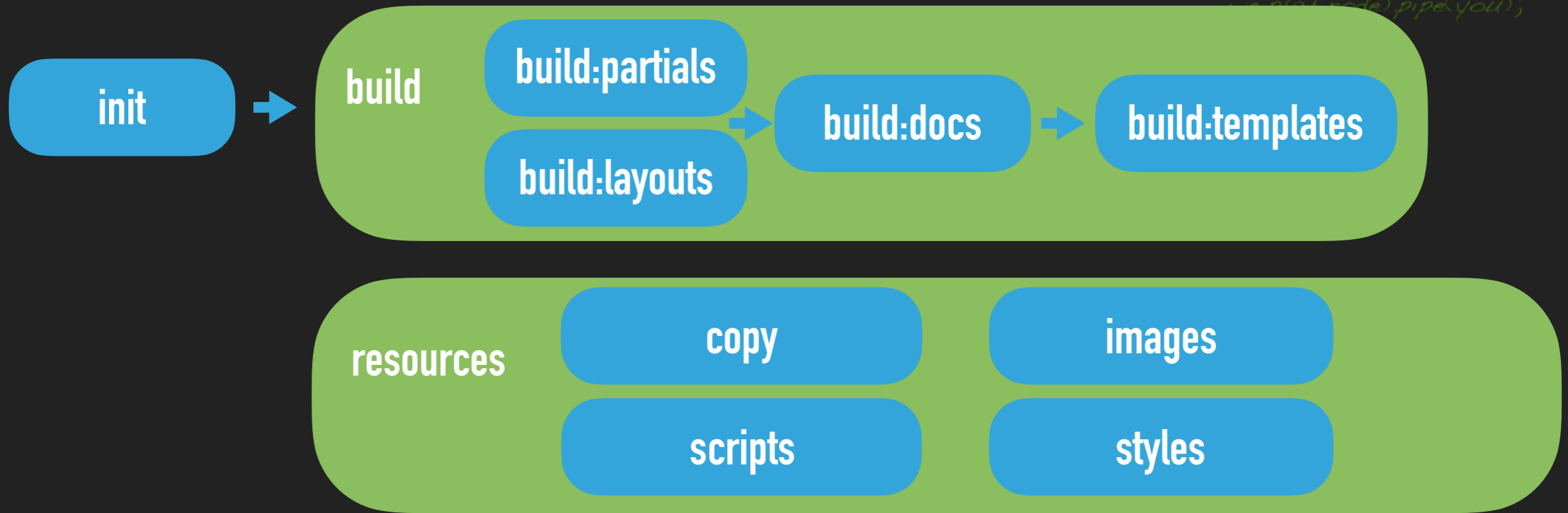
- ▶ 이미지 압축(gulp-cache, gulp-imagemin)



- ▶ gulp-sourcemaps
- ▶ gulp-jshint
- ▶ gulp-babel
- ▶ gulp-concat
- ▶ gulp-rename
- ▶ gulp-uglify



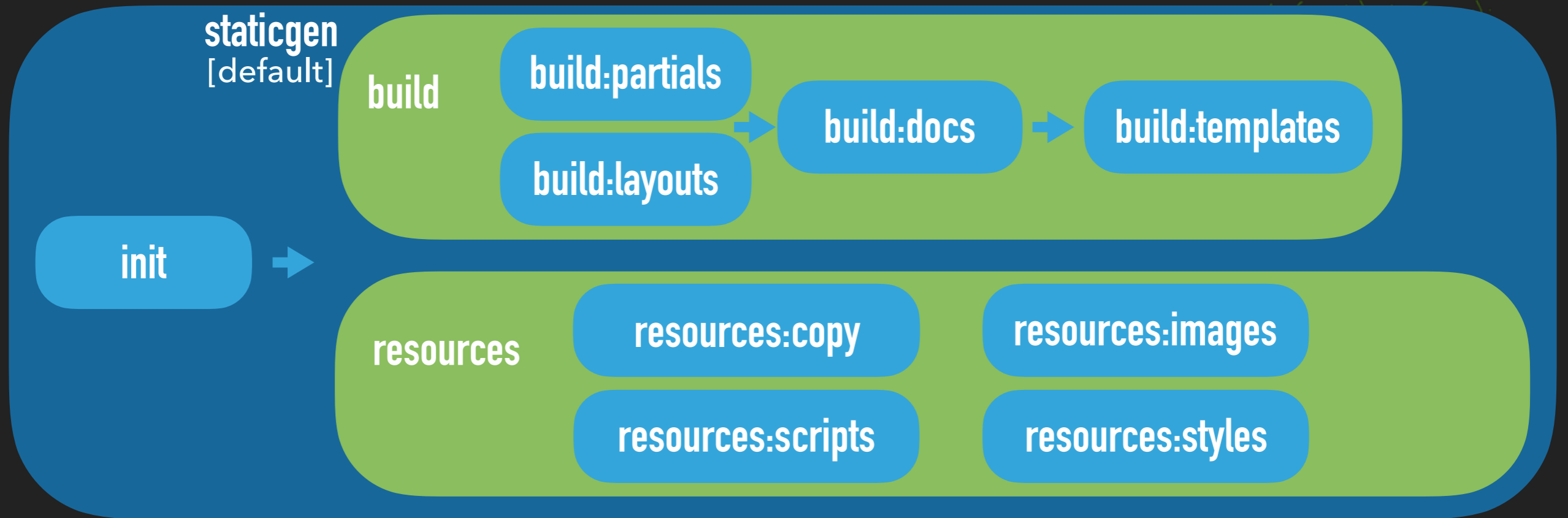
- ▶ gulp-sourcemaps
- ▶ gulp-autoprefixer
- ▶ gulp-less / gulp-sass
- ▶ gulp-concat
- ▶ gulp-rename
- ▶ gulp-minifyCss



```
gulp.task('resources', function(done) {  
  seq(['copy',  
    'images',  
    'scripts',  
    'styles'], // parallel  
  done);  
});
```



```
gulp.task('resources', function(done) {  
  seq(['resources:copy',  
    'resources:images',  
    'resources:scripts',  
    'resources:styles'],  
    done);  
});
```



```
gulp.task('staticgen', function(done) {  
  seq('init',  
    ['build', 'resources']  
    done);  
});  
  
gulp.task('default', 'staticgen');
```

watch

```
gulp.task('watch', function(done){
  // STYLES 파일이 변경되면 `resources:styles`을 실행한다.
  gulp.watch(global.STYLES, 'resources:styles');
  gulp.watch(global.SCRIPTS, 'resources:scripts');
  .....
});

gulp.task('resources:styles', function(){
  return gulp.src(global.STYLES)
    .pipe($.cached('resources:styles')) // 변경된 파일만 통과
    .pipe(.....);
});
```

watch

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

gulp.task('watch', function(done) {
  gulp.watch('**/*', function(event) {
    if( minimatch(event.path, local.STYLES) ) {
      stylesStream(event.path);
    } else if( ..... ) { ..... }
  });

  .....
});

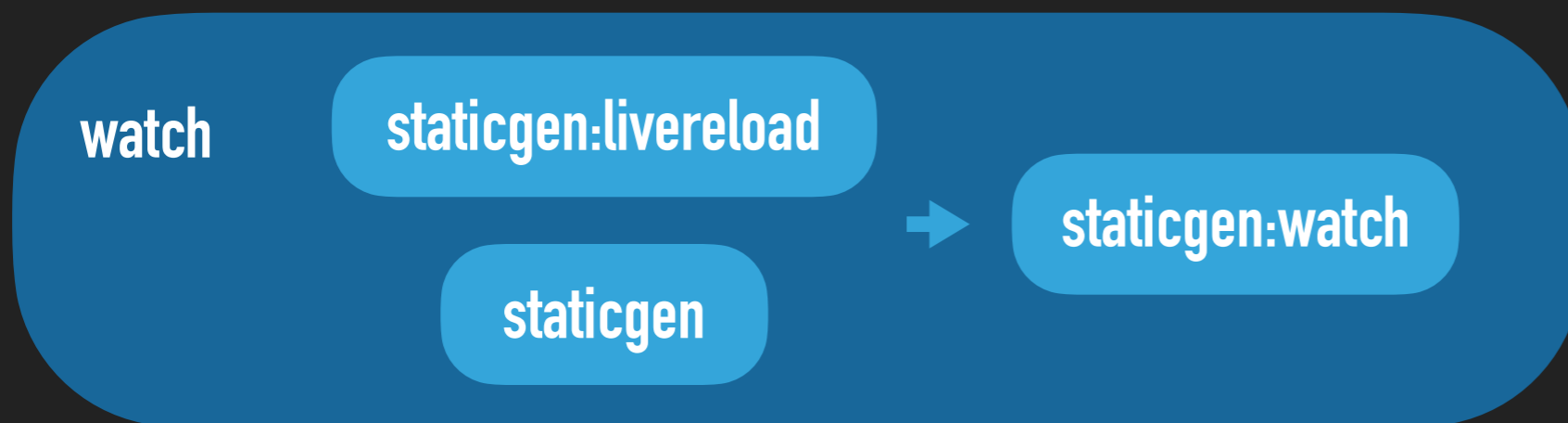
gulp.task('resources:styles', function() {
  return stylesStream(global.STYLES);
});
```


watch

livereload

```
gulp.task('livereload', function(done) {  
  $.connect.server({  
    livereload: true,  
    root: [global.DEST],  
    port: 9000  
  });  
});
```

```
gulp.task('docs', function(done) {  
  return gulp.src(global.DOCS)  
    .pipe(...)  
    .pipe($.connect.reload());  
});
```



```
gulp.task('watch', function(done) {  
  seq(['staticgen:livereload', 'staticgen'],  
    'staticgen:watch',  
    done);  
});
```



```
gulp.task('deploy', function(done){
  var deploy = require('gulp-gh-pages');
  var path = require('path');

  return gulp.src([path.join(global.DEST, '**/*'), '!**/*.map'])
    .pipe(deploy({
      remoteUrl: 'git@github.com:pismute/pismute.github.io.git',
      cacheDir: '.gh-pages',
      branch: 'master'
    }));
});
```



```
$ gulp help # gulp-task-listing
```

```
.....
```

Main Tasks

```
-----
```

```
build  
deploy  
help  
livereload  
resources  
watch
```

Sub Tasks

```
-----
```

```
build:docs  
build:layouts  
build:partials
```

```
.....
```

TASK 공유하기



```
gulp.task('staticgen', function(done){  
  seq('init',  
    ['build', 'resources']  
  done);  
});
```



```
gulp.task('staticgen', function(done){  
  seq('init',  
    ['build', 'resources']  
  done);  
});
```

```
var taskLibrary = require('my-task-library');
```



```
gulp.task('init', function(done) {  
  global.DOCS = 'my-docs/**/*.*md'  
  .....  
});
```



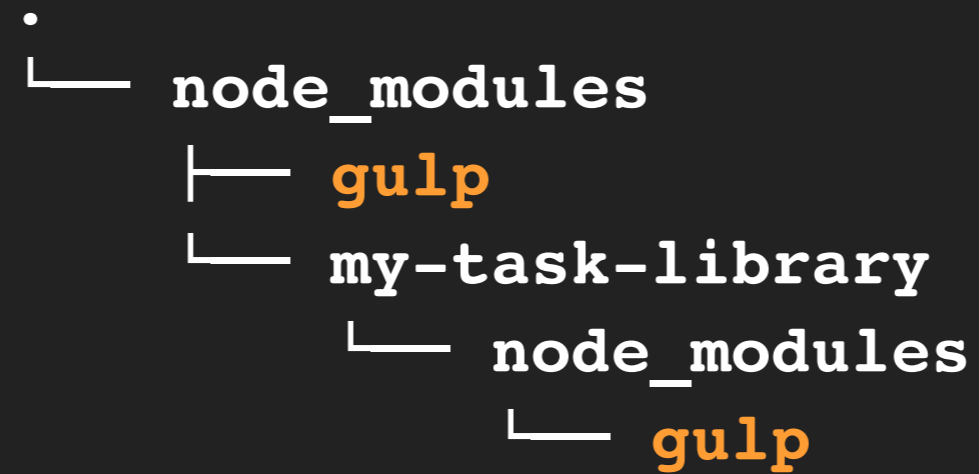
```
gulp.task('init', function(done) {  
  global.DOCS = 'my-docs/**/*.*asc'  
  .....  
});
```



```
var gulp = require('gulp');
var taskLibrary = require('my-task-library')

taskLibrary.init(gulp);

gulp.task('init', function(done) {
  global.DOCS = 'my-docs/**/* .md';
  .....
});
```

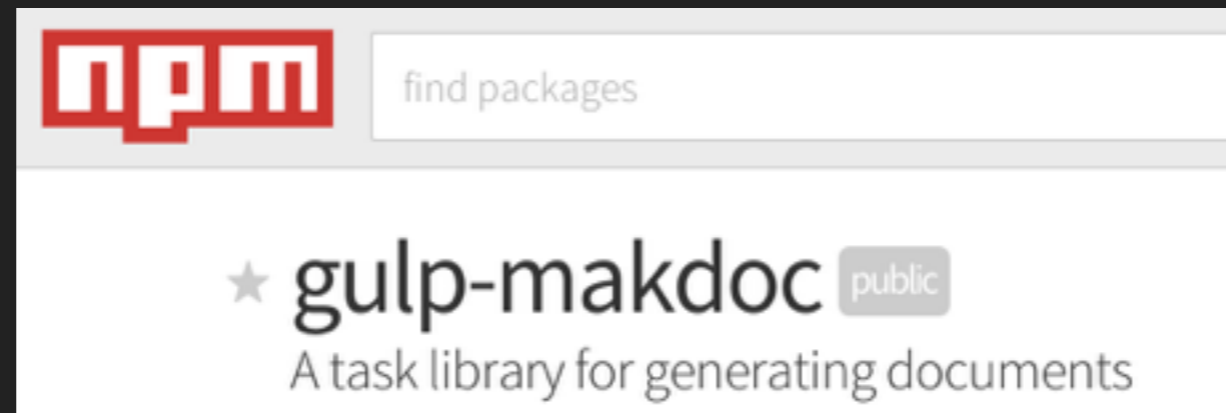


같은 것 같지만 **다른** 모듈!

```
var Handlebars = require('handlebars');
var taskLibrary = require('my-task-library')

taskLibrary.init(Handlebars);

Handlebars.registerHelper('square', function(n, options) {
  return n * n;
});
```



(<https://github.com/pismute/gulp-makdoc>)

```
$ npm install gulp-makdoc
```

```
var gulp = require('gulp');  
var Handlebars = require('handlebars');  
var makdoc = require('gulp-makdoc')(gulp, Handlebars);
```

- ▶ Gulp 좋아
 - ▶ pipe 재밋다
 - ▶ 비동기
 - ▶ 많이 빨라(grunt 보다)
 - ▶ 코드 짧아

ASCIIDOC

```
# H1
## H2
### H3
#### H4
##### H5
##### H6
```

Emphasis, aka italics, with `*asterisks*` or `_underscores_`.

Strong emphasis, aka bold, with `**asterisks**` or `__underscores__`.

Combined emphasis with `**asterisks and _underscores_**`.

Strikethrough uses two tildes. `~~Scratch this~~`

1. First ordered list item

2. Another item

* Unordered sub-list.

1. Actual numbers don't matter, just that it's a number

1. Ordered sub-list

4. And another item.

```
[I'm an inline-style link](https://www.google.com)
```

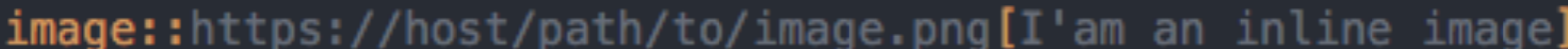
```
![I'am an inline image](https://host/path/to/image.png")
```

```
= H1
== H2
=== H3
==== H4
===== H5
===== H6
```

Emphasis, aka italics, with *underscores* or double underscores.
Strong emphasis, aka bold, with ***asterisks*** or **double underscores**.
Combined emphasis with ***asterisks and underscores***.
Strikethrough uses two tildes. [blue line-through]~~Scratch this~~.

```
. First ordered list item
. Another item
** Unordered sub-list.
. Actual numbers don't matter, just that it's a number
.. Ordered sub-list
. And another item.
```

```
https://www.google.com[I'm an inline-style link]
```

```
[I'am an inline image]
```


요약

- ▶ Markdown 처럼 사용할 수 있다.
- ▶ Attributes - 문서 속성이나 변수 정의 가능
- ▶ 문법강조 포함
- ▶ Include 문
- ▶ Tables
- ▶ Callouts
- ▶
- ▶ PDF 만들거라면 AsciiDoc

Callouts

MS-DOS directory listing.

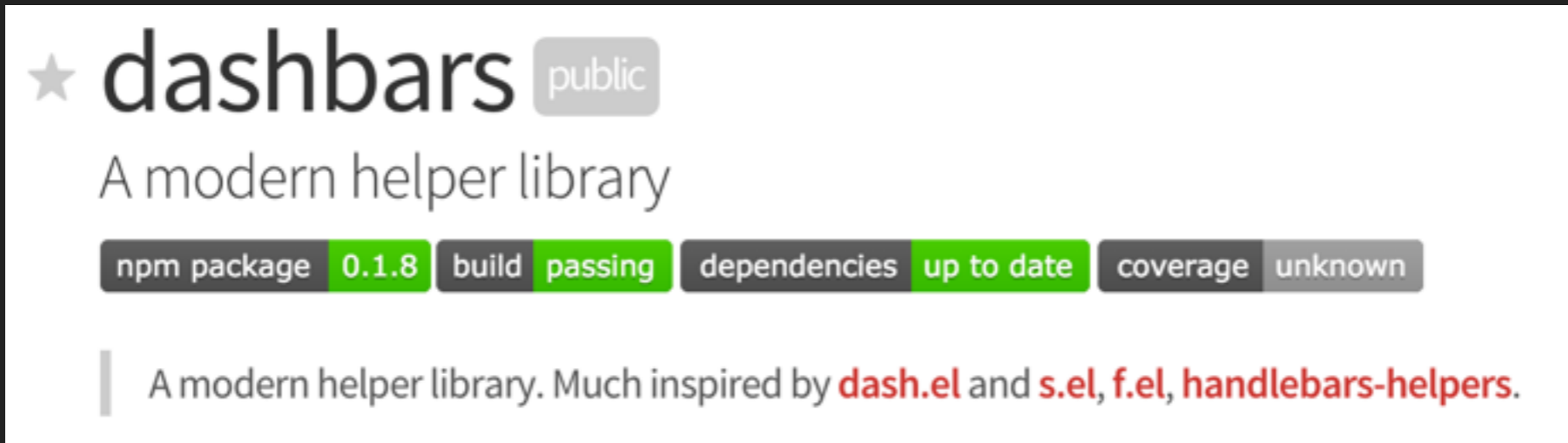
10/17/97	9:04	<DIR>	bin	
10/16/97	14:11	<DIR>	DOS	❶
10/16/97	14:40	<DIR>	Program Files	
10/16/97	14:46	<DIR>	TEMP	
10/17/97	9:04	<DIR>	tmp	
10/16/97	14:37	<DIR>	WINNT	
10/16/97	14:25		119 AUTOEXEC.BAT	❷
2/13/94	6:21		54,619 COMMAND.COM	❸
10/16/97	14:25		115 CONFIG.SYS	❹
11/16/97	17:17		61,865,984 pagefile.sys	
2/13/94	6:21		9,349 WINA20.386	❺

- ❶ This directory holds MS-DOS.
- ❷ ❸ ❹ System startup code for DOS.
- ❺ Some sort of Windows 3.1 hack.

HANDLEBARS.JS AND HELPER

Dashbars

Underscore를 Template에서..



★ dashbars public

A modern helper library

npm package 0.1.8 build passing dependencies up to date coverage unknown

A modern helper library. Much inspired by [dash.el](#) and [s.el](#), [f.el](#), [handlebars-helpers](#).

(<https://github.com/pismute/dashbars>)

감사합니다