gulp + sass で目指せ倍速コーディング

東区フロントエンド勉強会 2015年 第2回

gulpfile.js の分割管理と画像作成の効率化

第４章　gulpfile.js の分割管理
第５章　画像を自動で最適化しよう
第６章　default タスクで一括 watch
第７章　CSSスプライトを自動化しよう

エクスコード株式会社
http://excode.jp
フロントエンド エンジニア
末包 俊道（すえかね）

第４章　gulofile.js の分割管理

第４章　gulpfile.js の分割管理

第４章　gulofile.js の分割管理

1. 現状の確認
2. 設定ファイルを作ろう (config.js)
3. 設定ファイルを読み込もう
4. ここまでの動作確認をしよう
5. タスクを別ファイルで管理しよう
6. 動作確認をしよう

1. 現状の確認

1. 現状の確認

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');

gulp.task('sass', function () {
  gulp.src('./develop/sass/**/*.sass')
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest('./html/css'));
});

gulp.task('sass:watch', function () {
  gulp.watch('./develop/sass/**/*.sass',
['sass']);
});
```

1. 現状の確認

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');

gulp.task('sass', function () {
  gulp.src('./develop/sass/**/*.sass')
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest('./html/css'));
});

gulp.task('sass:watch', function () {
  gulp.watch('./develop/sass/**/*.sass',
['sass']);
});
```

案件に合わせて簡単に
変更できるようにしたい

1. 現状の確認

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');

gulp.task('sass', function () {
  gulp.src('./develop/sass/**/*.sass')
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest('./html/css'));
});

gulp.task('sass:watch', function () {
  gulp.watch('./develop/sass/**/*.sass',
['sass']);
});
```

案件に合わせて簡単に
変更できるようにしたい

設定ファイルを作成し、
一括管理しよう

2. 設定ファイルを作ろう (config.js)

2. 設定ファイルを作ろう(config.js)

Windows の方

gulpfile.js のあるディレクトリに移動

```
> cd Desktop\lesson
または
> cd C:\Users\ユーザー名\Desktop\lesson
```

config.js を作成します

```
> type nul > config.js
```

Mac の方

gulpfile.js のあるディレクトリに移動

```
$ cd ~/Desktop/lesson
```

config.js を作成します

```
$ touch config.js
```

2. 設定ファイルを作ろう(config.js)

```
lesson
   ├ .csscomb.json
   |
   ├ config.js ←今回追加した設定ファイル
   ├ develop
   |    └ sass
   |       └ style.sass
   |
   ├ gulpfile.js
   |
   ├ html
   |    ├ css
   |    |  └ style.css
   |    |
   |    └ index.html
   |
   ├ node_modules
   └ package.json
```

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ

// 納品用ディレクトリ

// データ受け渡し用のオブジェクトを作成

// Sass の設定
```

設定したいこと

・開発用ディレクトリのパス
・納品用ディレクトリのパス
・データ受け渡し用オブジェクト
・sass の設定

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ

// 納品用ディレクトリ

// データ受け渡し用のオブジェクトを作成

// Sass の設定
```

設定したいこと

・開発用ディレクトリのパス
  - ./develop

・納品用ディレクトリのパス
  - ./html

・データ受け渡し用オブジェクト

・Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src = './develop';

// 納品用ディレクトリ

// データ受け渡し用のオブジェクトを作成

// Sass の設定
```

※ src … source (元データ)

設定したいこと

- **開発用ディレクトリのパス**
  - **./develop**
- 納品用ディレクトリのパス
  - ./html
- データ受け渡し用オブジェクト
- Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src  = './develop';

// 納品用ディレクトリ
var dest = './html';

// データ受け渡し用のオブジェクトを作成

// Sass の設定
```

※ dest … destination (出力先)

設定したいこと

・開発用ディレクトリのパス
  - ./develop

・納品用ディレクトリのパス
  - ./html

・データ受け渡し用オブジェクト

・Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src  = './develop';

// 納品用ディレクトリ
var dest = './html';

// データ受け渡し用のオブジェクトを作成
module.exports = {

  // Sass の設定
};
```

設定したいこと

・開発用ディレクトリのパス
  - ./develop

・納品用ディレクトリのパス
  - ./html

・データ受け渡し用オブジェクト

・Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src  = './develop';

// 納品用ディレクトリ
var dest = './html';

// データ受け渡し用のオブジェクトを作成
module.exports = {

  // Sass の設定
};
```

例）オブジェクトの書式

```
object = {
  tanaka: {
    city: 'Fukuoka',
    phone: '090-xxxx-xxxx',
    age: 25,
    gender: 'male'
  }
};
```

例）田中さんの性別

```
object.tanaka.age

// male
```

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src  = './develop';

// 納品用ディレクトリ
var dest = './html';

// データ受け渡し用のオブジェクトを作成
module.exports = {

  // ディレクトリの情報を受け渡す
  src:  src,
  dest: dest,

  // Sass の設定
};
```

設定したいこと

・開発用ディレクトリのパス
  - ./develop

・納品用ディレクトリのパス
  - ./html

・データ受け渡し用オブジェクト

・Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

2. 設定ファイルを作ろう(config.js)

config.js

```
// 開発用ディレクトリ
var src  = './develop';

// 納品用ディレクトリ
var dest = './html';

// データ受け渡し用のオブジェクトを作成
module.exports = {

  // ディレクトリの情報を受け渡す
  src:  src,
  dest: dest,

  // Sass の設定
  sass: {
    src:  src + '/sass/**/*.sass',
    dest: dest + '/css',
  }
};
```

設定したいこと

・開発用ディレクトリのパス
  - ./develop

・納品用ディレクトリのパス
  - ./html

・データ受け渡し用オブジェクト

・Sass の設定
  - Sass ファイル（監視対象）
  - CSS 出力ディレクトリ

3. 設定ファイルを読み込もう

3. 設定ファイルを読み込もう

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src('./develop/sass/**/*.sass')
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest('./html/css'));
});

gulp.task('sass:watch', function () {
  gulp.watch('./develop/sass/**/*.sass',
['sass']);
});
```

設定ファイルを読み込みます

```
var conf = require('./config');
```

3. 設定ファイルを読み込もう

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

値を書き換えます

元データ（監視対象）の .sass ファイル

conf.sass.src

出力先ディレクトリ

conf.sass.dest

4. ここまでの動作確認をしよう

4. ここまでの動作確認をしよう

動作確認のため、watch モードに入ります

gulp sass:watch

Windows の方

gulp sass:watch タスクを実行してみます

```
> gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10
ms
```

Mac の方

gulp sass:watch タスクを実行してみます

```
$ gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after
10 ms
```

watch モードに入りました

4. ここまでの動作確認をしよう

前回までと同じように style.sass に変更を加え保存してみてください

style.sass に追記

```
h2
  color: blue
```

Windows の方

gulp sass:watch タスク

```
> gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10 ms
[15:38:40] Starting 'sass'...
[15:38:40] Finished 'sass' after 9.06 ms
```

Mac の方

gulp sass:watch タスク

```
$ gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10 ms
[15:38:40] Starting 'sass'...
[15:38:40] Finished 'sass' after 9.06 ms
```

CSSファイルが正しく更新されていれば完了です。

5. タスクを別ファイルで管理しよう

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

- **require-dir をインストール**
- requireDir を定義
  - ./gulp_task
- 外部タスク用ディレクトリを作成
  - ./gulp_task
- sass タスクを sass.js として保存
  - ./gulp_task/sass.js
- watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

一旦、watch から抜けて作業に入ります

Windows の方

ctrl + c を押し、一旦タスクを終了します

```
[13:09:14] Starting 'sass'...
[13:09:14] Finished 'sass' after 980 µs
^C
>
```

Mac の方

ctrl + c を押し、一旦タスクを終了します

```
[13:09:14] Starting 'sass'...
[13:09:14] Finished 'sass' after 980 µs
^C
$
```

5. タスクを別ファイルで管理しよう

require-dir

https://www.npmjs.com/package/require-dir

指定したディレクトリ内の .js ファイルを自動で読み込みます

5. タスクを別ファイルで管理しよう

require-dir をインストールします

Windows の方

require-dir を追加

```
> npm install require-dir --save-dev
```

Mac の方

require-dir を追加

```
$ npm install require-dir --save-dev
```

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

- **requireDir を定義**
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

タスク用 gulp_task ディレクトリを作成

Windows の方

gulp_task ディレクトリを作成

```
> mkdir gulp_task
```

Mac の方

gulp_task ディレクトリを作成

```
$ mkdir gulp_task
```

ついでに sass.js watch.js も作成します

gulp_task 内に sass.js ファイルを作成

```
> type nul gulp_task\sass.js
> type nul gulp_task\watch.js
```

gulp_task 内に sass.js ファイルを作成

```
$ touch gulp_task/sass.js
$ touch gulp_task/watch.js
```

5. タスクを別ファイルで管理しよう

```
lesson
   ├ .csscomb.json
   |
   ├ config.js
   ├ develop
   |    └ sass
   |       └ style.sass
   |
   ├ gulp_task ←今回追加したタスク用ディレクトリ
   |    ├ sass.js
   |    └ watch.js
   |
   ├ gulpfile.js
   |
   ├ html
   |    ├ css
   |    |  └ style.css
   |    |
   |    └ index.html
   |
   ├ node_modules
   └ package.json
```

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');

var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('./config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

sass.js (./gulp_task/sass.js)

```
var gulp = require('gulp');
var sass = require('gulp-sass');
var csscomb = require('gulp-csscomb');
var autoprefixer = require('gulp-autoprefixer');
var plumber = require('gulp-plumber');
var conf = require('../config');

gulp.task('sass', function () {
  gulp.src(conf.sass.src)
    .pipe(plumber())
    .pipe(sass().on('error', sass.logError))
    .pipe(csscomb())
    .pipe(autoprefixer({
      browsers:['safari 5', 'ie 8', 'ie 9',
                'ie 10', 'ie 11', 'opera 12.1',
                'firefox 14', 'ios 6',
                'android 2.1'],
      cascade: false
    }))
    .pipe(gulp.dest(conf.sass.dest));
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

5. タスクを別ファイルで管理しよう

gulpfile.js

```
'use strict';

var requireDir = require('require-dir');
var dir = requireDir('./gulp_task');
```

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・require-dir をインストール

・requireDir を定義
  - ./gulp_task

・外部タスク用ディレクトリを作成
  - ./gulp_task

・sass タスクを sass.js として保存
  - ./gulp_task/sass.js

・watch タスクを watch.js として保存
  - ./gulp_task/watch.js

6. 動作確認をしよう

6. 動作確認をしよう

動作確認のため、watch モードに入ります

gulp sass:watch

Windows の方

gulp sass:watch タスクを実行してみます

```
> gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10
ms
```

Mac の方

gulp sass:watch タスクを実行してみます

```
$ gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after
10 ms
```

watch モードに入りました

6. 動作確認をしよう

style.sass に変更を加え保存してみてください

style.sass に追記

```
h2
  color: pink
```

Windows の方

gulp sass:watch タスク

```
> gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10 ms
[15:38:40] Starting 'sass'...
[15:38:40] Finished 'sass' after 9.06 ms
```

Mac の方

gulp sass:watch タスク

```
$ gulp sass:watch
[15:38:32] Starting 'sass:watch'...
[15:38:32] Finished 'sass:watch' after 10 ms
[15:38:40] Starting 'sass'...
[15:38:40] Finished 'sass' after 9.06 ms
```

CSSファイルが正しく更新されていれば完了です。

第５章　画像を自動で最適化しよう

第５章　画像を自動で最適化しよう

1. 概要
2. タスクを作ろう
3. 動作確認をしよう

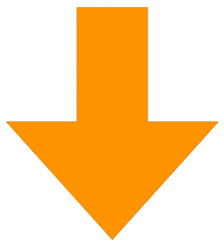
1. 概要

作業用のディレクトリに保存

JPEG (93KB)

JPEG (65KB)

PNG (92KB)



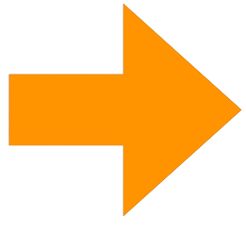
作業用ディレクトリ

最適化された画像が書き出される

JPEG (70KB)

JPEG (48KB)

PNG (25KB)



最適化(納品用)ディレクトリ

2. タスクを作ろう

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

image.js (./gulp_task/image.js)

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
  - ./gulp_task/watch.js

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

image.js (./gulp_task/image.js)

```
```

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
  - ./gulp_task/watch.js

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

2. タスクを作ろう

gulp-imagemin

https://www.npmjs.com/package/gulp-imagemin

OptiPNG を使い画像を圧縮してくれます

imagemin-pngquant

https://www.npmjs.com/package/imagemin-pngquant

OptiPNG よりも高い圧縮率で PNG 画像を圧縮してくれます

gulp-changed

https://www.npmjs.com/package/gulp-changed

変更があったファイルのみを gulp のストリームに流します

gulp-watch

https://www.npmjs.com/package/gulp-watch

ファイルの追加も監視対象に含めます

2. タスクを作ろう

必要なプラグインをインストールします

watch に入っている方は抜けてください

Windows の方

gulp-imagemin を追加

```
> npm install gulp-imagemin --save-dev
```

imagemin-pngquant を追加

```
> npm install imagemin-pngquant --save-
dev
```

gulp-changed を追加

```
> npm install gulp-changed --save-dev
```

gulp-watch を追加

```
> npm install gulp-watch --save-dev
```

Mac の方

gulp-imagemin を追加

```
$ npm install gulp-imagemin --save-dev
```

imagemin-pngquant を追加

```
$ npm install imagemin-pngquant --save-
dev
```

gulp-changed を追加

```
$ npm install gulp-changed --save-dev
```

gulp-watch を追加

```
$ npm install gulp-watch --save-dev
```

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

image.js (./gulp_task/image.js)

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./images

・設定ファイルを編集
- ./config.js

・image タスクを image.js として保存
- ./gulp_task/image.js

・gulp.watch を watch に変更
- ./gulp_task/watch.js

・watch タスクを watch.js に追記
- ./gulp_task/watch.js

2. タスクを作ろう

Windows の方

images ディレクトリを作成

```
> mkdir develop\images
```

Mac の方

images ディレクトリを作成

```
$ mkdir develop/images
```

ついでに image.js も作成します

gulp_task 内に sass.js ファイルを作成

```
> type nul gulp_task\image.js
```

gulp_task 内に sass.js ファイルを作成

```
$ touch gulp_task/image.js
```

2. タスクを作ろう

```
lesson
   ├ .csscomb.json
   |
   ├ config.js
   ├ develop
   |    ├ images ←今回追加した作業用ディレクトリ
   |    └ sass
   |       └ style.sass
   |
   ├ gulp_task
   |    ├ image.js
   |    ├ sass.js
   |    └ watch.js
   |
   ├ gulpfile.js
   |
   ├ html
   |    ├ css
   |    |  └ style.css
   |    ├ images ←タスク実行時に自動で生成されます
   |    └ index.html
   |
   ├ node_modules
   └ package.json
```

2. タスクを作ろう

config.js

```
var src  = './develop';
var dest = './html';

module.exports = {

  src:  src,
  dest: dest,

  sass: {
    src:  src + '/sass/**/*.sass',
    dest: dest + '/css',
  },

  image: {
    src:  src + '/images/**/*',
    dest: dest + '/images',
  },

  sprite: {
    src:  src + '/sprite/**/*',
    dest: dest + '/sprite',
  },
};
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
  - ./gulp_task/watch.js

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

2. タスクを作ろう

image.js (./gulp_task/image.js)

```
var gulp = require('gulp');
var imagemin = require('gulp-imagemin');
var pngquant = require('imagemin-pngquant');
var changed  = require('gulp-changed');
var plumber  = require('gulp-plumber');
var conf = require('../config');


gulp.task('image', function () {
  return gulp.src(conf.image.src)
    .pipe(plumber())
    .pipe(changed(conf.image.dest))
    .pipe(imagemin({
      progressive: true,
      svgoPlugins: [{removeViewBox: false}],
      use: [pngquant()]
    }))
    .pipe(gulp.dest(conf.image.dest));
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
  - ./gulp_task/watch.js

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

2. タスクを作ろう

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var watch = require('gulp-watch');
var conf = require('../config');

gulp.task('sass:watch', function () {
  // gulp.watch(conf.sass.src, ['sass']);
  watch(conf.sass.src, function () {
    gulp.start(['sass']);
  });
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
** - ./gulp_task/watch.js**

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

2. タスクを作ろう

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var watch = require('gulp-watch');
var conf = require('../config');

gulp.task('sass:watch', function () {
  // gulp.watch(conf.sass.src, ['sass']);
  watch(conf.sass.src, function () {
    gulp.start(['sass']);
  });
});

gulp.task('image:watch', function () {
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./images

・設定ファイルを編集
  - ./config.js

・image タスクを image.js として保存
  - ./gulp_task/image.js

・gulp.watch を watch に変更
  - ./gulp_task/watch.js

・watch タスクを watch.js に追記
  - ./gulp_task/watch.js

3. 動作確認をしよう

3. 動作確認をしよう

動作確認のため、watch モードに入ります

gulp image:watch

Windows の方

gulp sass:watch タスクを実行してみます

```
> gulp image:watch
[15:38:32] Starting 'image:watch'...
[15:38:32] Finished 'image:watch' after
10 ms
```

Mac の方

gulp sass:watch タスクを実行してみます

```
$ gulp image:watch
[15:38:32] Starting 'image:watch'...
[15:38:32] Finished 'image:watch' after
10 ms
```

watch モードに入りました

3. 動作確認をしよう

wolf.png (92KB)

./develop/images

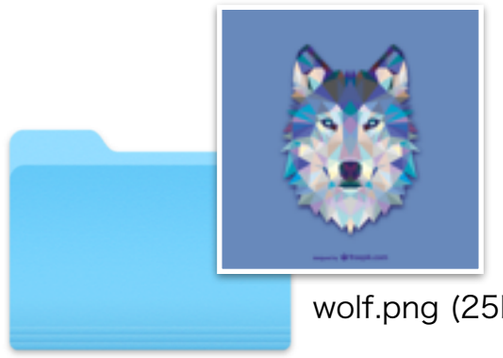
./develop/images に画像を保存

Windows の方

```
> gulp image:watch
[15:38:32] Starting 'image:watch'...
[15:38:32] Finished 'image:watch' after 10 ms
[15:38:40] Starting 'image'…
[15:38:40] gulp-imagemin: Minified 1 image (saved 67.23 kB - 72.7%)
[15:38:40] Finished 'image' after 9.06 ms
```

Mac の方

```
$ gulp image:watch
[15:38:32] Starting 'image:watch'...
[15:38:32] Finished 'image:watch' after 10 ms
[15:38:40] Starting 'image'…
[15:38:40] gulp-imagemin: Minified 1 image (saved 67.23 kB - 72.7%)
[15:38:40] Finished 'image' after 9.06 ms
```



wolf.png (25KB)

./html/images

./html/images に最適化された画像が生成されていればOK

第６章　default タスクで一括 watch

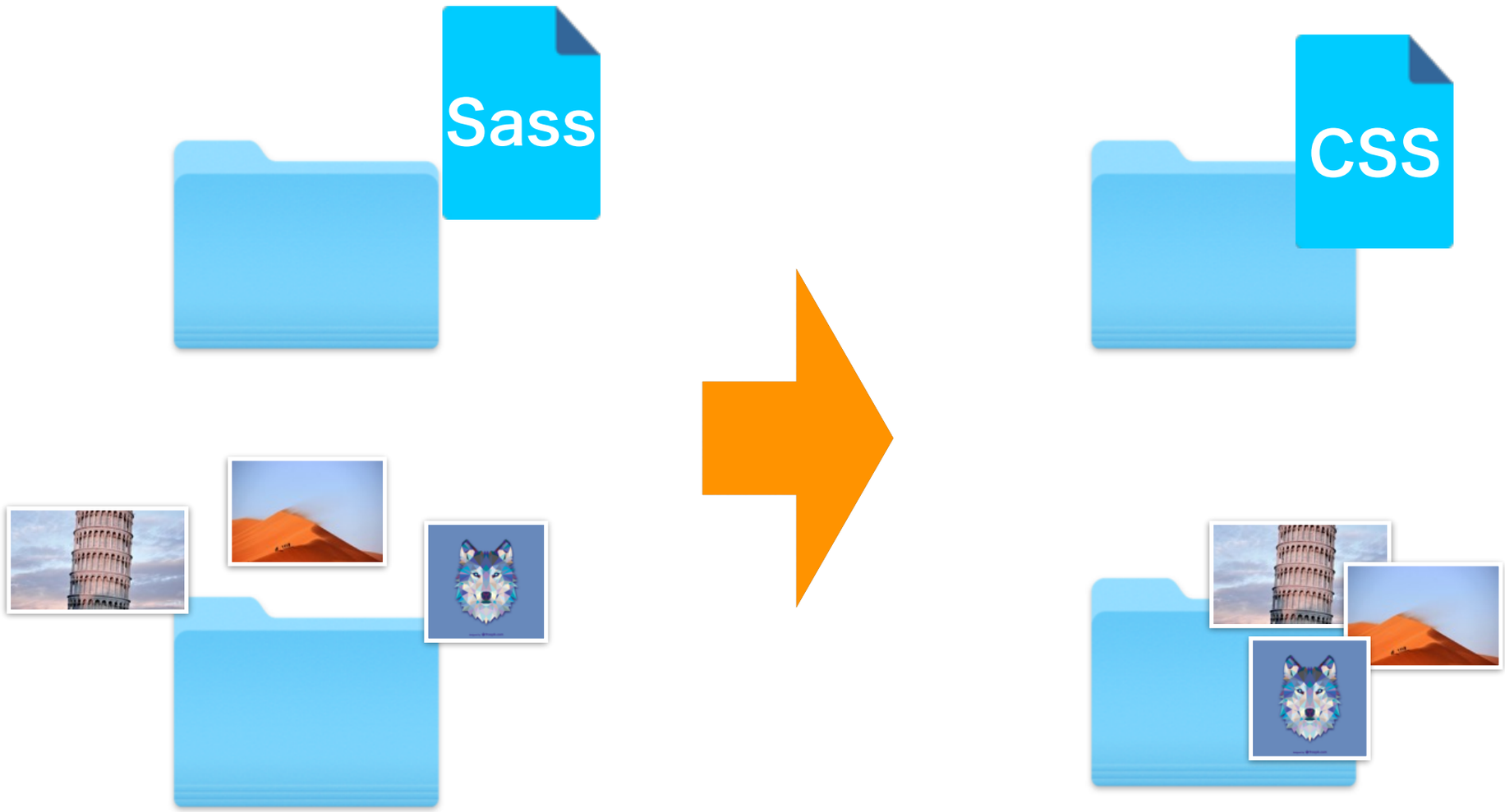
第６章　default タスクで一括 watch

1. 概要
2. タスクを作ろう
3. 動作確認をしよう

1. 概要

Sass も画像も全部監視対象にしたい

gulp



2. タスクを作ろう

2. タスクを作ろう

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var watch = require('gulp-watch');
var conf = require('../config');

gulp.task('sass:watch', function () {
  // gulp.watch(conf.sass.src, ['sass']);
  watch(conf.sass.src, function () {
    gulp.start(['sass']);
  });
});

gulp.task('image:watch', function () {
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
});

gulp.task('default', function() {
  watch(conf.sass.src, function () {
    gulp.start(['sass']);
  });
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
});
```

作業手順　watch に入っている方は抜けてください

- **default タスクを作成し、中身をコピペ**

3. 動作確認をしよう

3. 動作確認をしよう

動作確認のため、watch モードに入ります

gulp

Windows の方

gulp タスクを実行してみます

```
> gulp
[15:38:32] Starting 'default'...
[15:38:32] Finished 'default' after 10 ms
```

Mac の方

gulp タスクを実行してみます

```
$ gulp
[15:38:32] Starting 'default'...
[15:38:32] Finished 'default' after 10
ms
```

watch モードに入りました

3. 動作確認をしよう

sass を変更したり、画像を追加したりしてみてください

Windows の方

gulp sass:watch タスク

```
> gulp sass:watch
[16:51:18] Starting 'default'...
[16:51:18] Finished 'default' after 13 ms
[16:51:28] Starting 'image'...
[16:51:29] gulp-imagemin: Minified 1 image
(saved 67.23 kB - 72.7%)
[16:51:29] Finished 'image' after 708 ms
[16:51:36] Starting 'sass'...
[16:51:36] Finished 'sass' after 18 ms
```

Mac の方

gulp sass:watch タスク

```
$ gulp sass:watch
[16:51:18] Starting 'default'...
[16:51:18] Finished 'default' after 13 ms
[16:51:28] Starting 'image'...
[16:51:29] gulp-imagemin: Minified 1 image
(saved 67.23 kB - 72.7%)
[16:51:29] Finished 'image' after 708 ms
[16:51:36] Starting 'sass'...
[16:51:36] Finished 'sass' after 18 ms
```

意図した通り動作していればOKです

第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Windows 環境でも構築できる Retina対応のCSS スプライト自動化については Sprity を用い別資料を用意します。

第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Windows 環境でも構築できる Retina対応のCSS スプライト自動化については Sprity を用い別資料を用意します。

第７章　CSSスプライトを自動化しよう

1. 概要
2. タスクを作ろう
3. 動作確認をしよう
4. Sassファイルの使い方

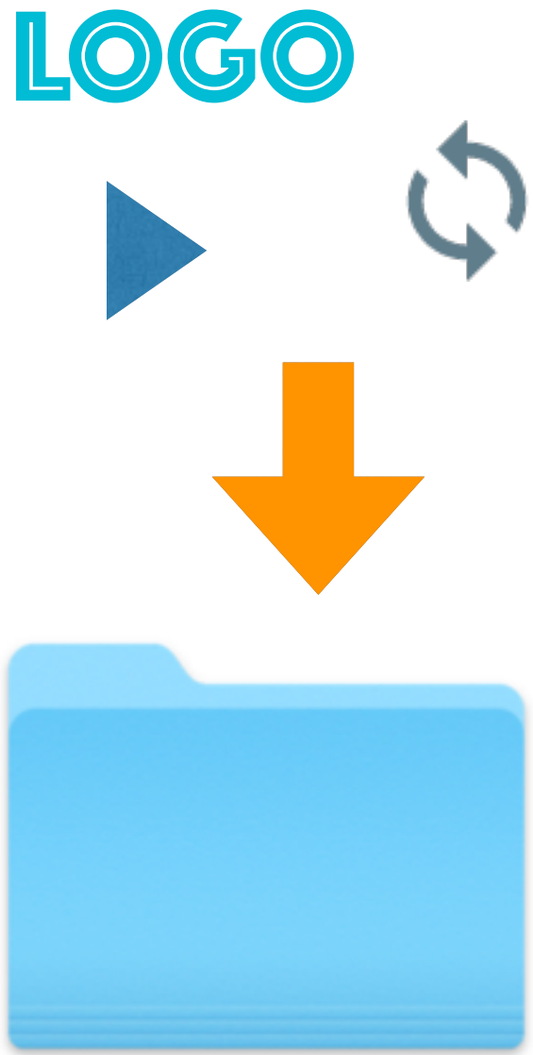
第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Windows 環境でも構築できる Retina対応のCSS スプライト自動化については Sprity を用い別資料を用意します。

1. 概要

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Windows 環境でも構築できる Retina対応のCSS スプライト自動化については Sprity を用い別資料を用意します。

作業用のディレクトリに保存



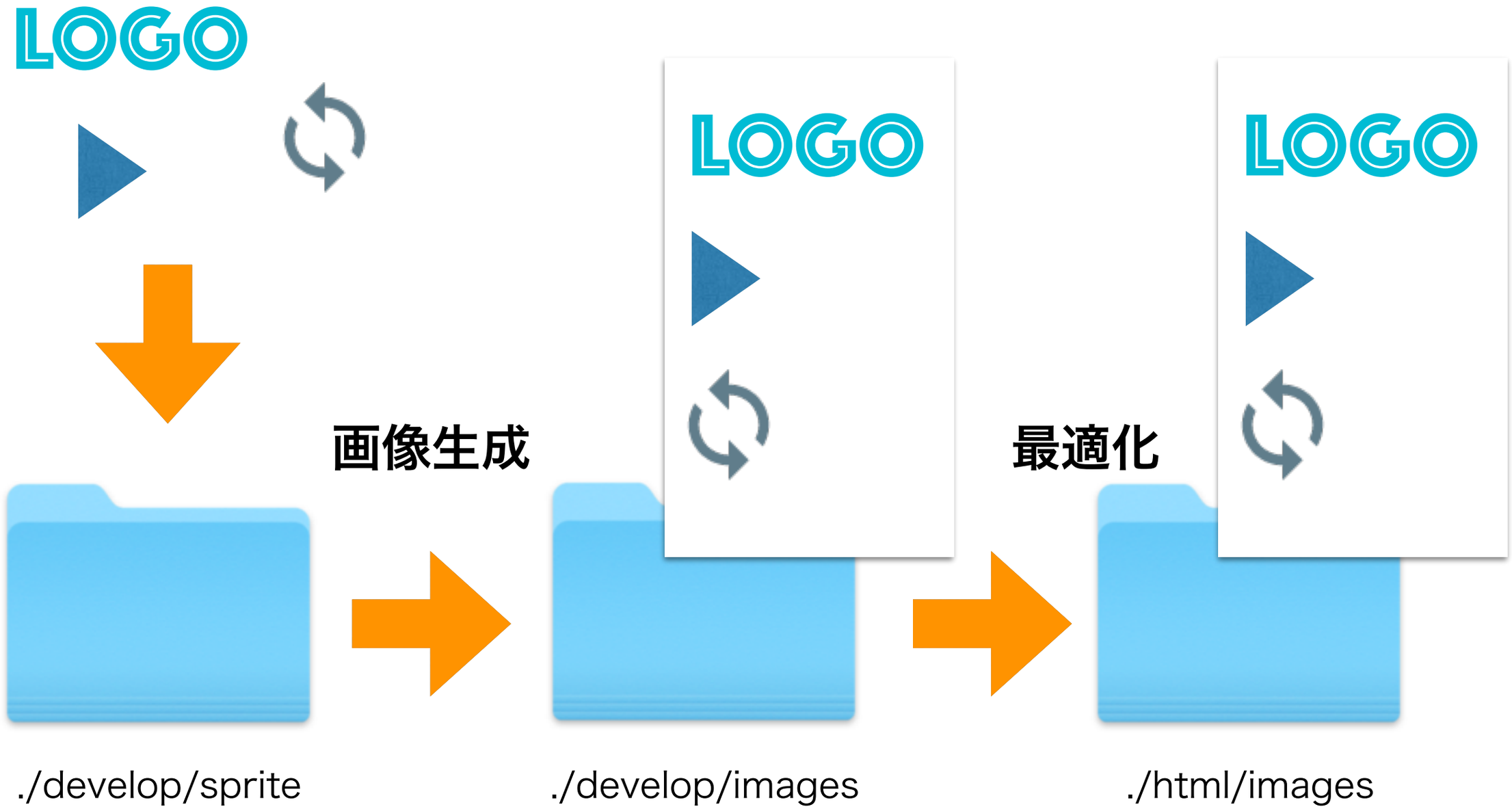
./develop/sprite

CSSスプライトが生成される

./html/images

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Windows 環境でも構築できる Retina対応のCSS スプライト自動化については Sprity を用い別資料を用意します。

作業用のディレクトリに保存 → 最適化したCSSスプライト画像を生成



第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

sprite.js (./gulp_task/sprite.js)

```
```

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});

…（省略）
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./sprite

・設定ファイルを編集
  - ./config.js

・ sprite タスクを sprite.js として保存
  - ./gulp_task/sprite.js

・watch タスクに追記
  - ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

sprite.js (./gulp_task/sprite.js)

```
```

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});

…（省略）
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./sprite

・設定ファイルを編集
- ./config.js

・ sprite タスクを sprite.js として保存
- ./gulp_task/sprite.js

・watch タスクに追記
- ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

css-sprite

https://www.npmjs.com/package/css-sprite

ディレクトリに追加された画像をスプライト画像とSassを書き出します

gulp-if

https://www.npmjs.com/package/gulp-if

条件分岐を使えるようになります

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

Windows の方

css-sprite を追加

```
> npm install css-sprite –save-dev -g
```

gulp-if を追加

```
> npm install gulp-if --save-dev
```

Mac の方

css-sprite を追加

```
$ npm install css-sprite –save-dev -g
```

gulp-if を追加

```
$ npm install gulp-if --save-dev
```

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

config.js

```
// 開発用ディレクトリ
var src  = './develop';

…（省略）
```

sprite.js (./gulp_task/sprite.js)

```
```

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var conf = require('../config');

gulp.task('sass:watch', function () {
  gulp.watch(conf.sass.src, ['sass']);
});

…（省略）
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./sprite

・設定ファイルを編集
- ./config.js

・ sprite タスクを sprite.js として保存
- ./gulp_task/sprite.js

・watch タスクに追記
- ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

Mac の方

sprite ディレクトリを作成

```
$ mkdir develop/sprite
```

Windows の方

sprite ディレクトリを作成

```
> mkdir develop\sprite
```

ついでに sprite.js も作成します

gulp_task 内に sprite.js ファイルを作成

```
$ touch gulp_task/sprite.js
```

gulp_task 内に sprite.js ファイルを作成

```
> type nul gulp_task\sprite.js
```

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

```
lesson
   ├ .csscomb.json
   |
   ├ config.js
   ├ develop
   |    ├ images
   |    ├ sass
   |    |  └ style.sass
   |    └ sprite ←今回追加した作業用ディレクトリ
   |
   ├ gulp_task
   |    ├ image.js
   |    ├ sass.js
   |    ├ sprite.js
   |    └ watch.js
   |
   ├ gulpfile.js
   |
   ├ html
   |    ├ css
   |    |  └ style.css
   |    ├ images ←タスク実行時に自動で生成されます
   |    └ index.html
   |
   ├ node_modules
   └ package.json
```

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

config.js

```
var src  = './develop';
var dest = './html';

module.exports = {

  src:  src,
  dest: dest,

  sass: {
    src:  src + '/sass/**/*.sass',
    dest: dest + '/css',
  },

  image: {
    src:  src + '/images/**/*',
    dest: dest + '/images',
  },

  sprite: {
    src:        src +'/sprite',
    dest:       src +'/images',   // スプライト画像出力先
    sass:       src +'/sass',     // Sassディレクトリ
    style:      '_sprite.sass',   // Sassファイル名
    name:       'my-sprite',      // 画像ファイル名
    prefix:     'sprite',         // mixinプレフィクス
    cssPath:    '../images/',     // CSS内のパス
    processor:  'sass',           // sass/scssほか
    retina:     true,             // Retina生成 true/false
  },
};
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
  - ./sprite

・設定ファイルを編集
  - ./config.js

・ sprite タスクを sprite.js として保存
  - ./gulp_task/sprite.js

・watch タスクに追記
  - ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

sprite.js (./gulp_task/sprite.js)

```
var gulp = require('gulp');
var gulpif = require('gulp-if');
var sprite = require('css-sprite').stream;

// generate sprite.png and _sprite.scss
gulp.task('sprites', function () {
  return gulp.src('./src/img/*.png')
    .pipe(sprite({
      name: 'sprite',
      style: '_sprite.scss',
      cssPath: './img',
      processor: 'scss'
    }))
    .pipe(gulpif('*.png', gulp.dest('./dist/
img/'), gulp.dest('./dist/scss/')))
});
```

Usage with Gulp
// generate sprite.png and _sprite.scss
の内容をコピペします

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./sprite

・設定ファイルを編集
- ./config.js

・ **sprite タスクを sprite.js として保存**
- ./gulp_task/sprite.js

・watch タスクに追記
- ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

sprite.js (./gulp_task/sprite.js)

```
var gulp = require('gulp');
var gulpif = require('gulp-if');
var sprite = require('css-sprite').stream;
var plumber  = require('gulp-plumber');
var conf = require('../config');

gulp.task('sprites', function () {
  return gulp.src(conf.sprite.src)
    .pipe(plumber())
    .pipe(sprite({
      name: conf.sprite.name,
      style: conf.sprite.style,
      cssPath: conf.sprite.cssPath,
      processor: conf.sprite.processor,
      retina: conf.sprite.retina,
      prefix: conf.sprite.prefix,
    }))
    .pipe(gulpif('*.png',
      gulp.dest(conf.sprite.spriteDest),
      gulp.dest(conf.sprite.sass)));
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./sprite

・設定ファイルを編集
- ./config.js

• sprite タスクを sprite.js として保存
- ./gulp_task/sprite.js

・watch タスクに追記
- ./gulp_task/watch.js

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

2. タスクを作ろう

watch.js (./gulp_task/watch.js)

```
var gulp = require('gulp');
var watch = require('gulp-watch');
var conf = require('../config');

gulp.task('sass:watch', function () {
  … (省略) …
});

gulp.task('image:watch', function () {
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
});

gulp.task('sprite:watch', function () {
  watch(conf.sprite.src, function () {
    gulp.start(['sprite']);
  });
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
});

gulp.task('default', function() {
  watch(conf.sass.src, function () {
    gulp.start(['sass']);
  });
  watch(conf.image.src, function () {
    gulp.start(['image']);
  });
  watch(conf.sprite.src, function () {
    gulp.start(['sprite']);
  });
});
```

作業手順

・必要なプラグインをインストール

・作業用ディレクトリを作成
- ./sprite

・設定ファイルを編集
- ./config.js

・ sprite タスクを sprite.js として保存
- ./gulp_task/sprite.js

・watch タスクに追記
- ./gulp_task/watch.js

第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

3. 動作確認をしよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

3. 動作確認をしよう

動作確認のため、watch モードに入ります

gulp

Windows の方

gulp タスクを実行してみます

```
> gulp
[15:38:32] Starting 'default'...
[15:38:32] Finished 'default' after 10 ms
```

Mac の方

gulp タスクを実行してみます

```
$ gulp
[15:38:32] Starting 'default'...
[15:38:32] Finished 'default' after 10
ms
```

watch モードに入りました

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

3. 動作確認をしよう

sprite ディレクトリに画像を追加してみてください

Windows の方

gulp sass:watch タスク

```
> gulp sass:watch
[22:34:19] Starting 'default'...
[22:34:19] Finished 'default' after 10 ms
[22:34:42] Starting 'sprite'...
[22:34:42] Finished 'sprite' after 155 ms
[22:34:42] Starting 'image'...
[22:34:42] Starting 'sass'...
[22:34:42] Finished 'sass' after 8.27 ms
[22:34:45] gulp-imagemin: Minified 2 images
(saved 203.96 kB - 74.5%)
[22:34:45] Finished 'image' after 2.28 s
```

Mac の方

gulp sass:watch タスク

```
$ gulp sass:watch
[22:34:19] Starting 'default'...
[22:34:19] Finished 'default' after 10 ms
[22:34:42] Starting 'sprite'...
[22:34:42] Finished 'sprite' after 155 ms
[22:34:42] Starting 'image'...
[22:34:42] Starting 'sass'...
[22:34:42] Finished 'sass' after 8.27 ms
[22:34:45] gulp-imagemin: Minified 2 images
(saved 203.96 kB - 74.5%)
[22:34:45] Finished 'image' after 2.28 s
```

意図した通り動作していればOKです

第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

第７章　CSSスプライトを自動化しよう

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

my-sprite.png (./html/images/my-sprite.png)

logo.png
↓
$logo

right.png
↓
$right

cycle.png
↓
$cycle

保存したファイル名が Sass 内の変数名になります

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
@import sprite

.logo-mark
  +sprite($logo)

h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

my-sprite.png (./html/images/my-sprite.png)

logo.png
↓
$logo

right.png
↓
$right

cycle.png
↓
$cycle

保存したファイル名が Sass 内の変数名になります

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
@import sprite

.logo-mark
  +sprite($logo)

h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

style.css (./home/css/style.sass)

```
.sprite {
  background-image: url('../images/my-
sprite.png');
}

@media (min--moz-device-pixel-ratio: 1.5), (-
webkit-min-device-pixel-ratio: 1.5), (min-device-
pixel-ratio: 1.5), (min-resolution: 1.5dppx) {
  .sprite {
    background-image: url('../images/my-
sprite@2x.png');
    -webkit-background-size: 204px 476px;
    background-size: 204px 476px;
  }
}

.logo-mark {
  background-position: -2px -274px;
  background-repeat: no-repeat;
  overflow: hidden;
  display: block;
  width: 200px;
  height: 200px;
}

h1 {
  …（以下省略）…
```

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
@import sprite

.logo-mark
  +sprite($logo)

h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

style.css (./home/css/style.sass)

```
.sprite {
  background-image: url('../images/my-
sprite.png');
}

@media (min--moz-device-pixel-ratio: 1.5), (-
webkit-min-device-pixel-ratio: 1.5), (min-device-
pixel-ratio: 1.5), (min-resolution: 1.5dppx) {
  .sprite {
    background-image: url('../images/my-
sprite@2x.png');
    -webkit-background-size: 204px 476px;
  }
}

.logo-mark {
  background-position: -2px -274px;
  background-repeat: no-repeat;
  overflow: hidden;
  display: block;
  width: 200px;
  height: 200px;
}

h1 {
  … (以下省略) …
```

この状態では background 関連の指定が効いていません

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
@import sprite

.logo-mark
  @extend .sprite
  +sprite($logo)

h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

生成されれいる .sprite クラスを読み込みます

※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

style.sass (./develop/sass/style.sass)

```
@import sprite

.logo-mark
  @extend .sprite
  +sprite($logo)

h1
  margin: 0 auto
  a
    color: red
  &.span
    transform: rotate(-14deg)
    background-size: 20px 20px
h2
  color: pink
```

style.css (./home/css/style.sass)

```
.sprite, .logo-mark {
  background-image: url('../images/my-
sprite.png');
}

@media (min--moz-device-pixel-ratio: 1.5), (-
webkit-min-device-pixel-ratio: 1.5), (min-device-
pixel-ratio: 1.5), (min-resolution: 1.5dppx) {
  .sprite {
    background-image: url('../images/my-
sprite@2x.png');
    -webkit-background-size: 204px 476px;
    background-size: 204px 476px;
  }
}

.logo-mark {
  background-position: -2px -274px;
  background-repeat: no-repeat;
  overflow: hidden;
  display: block;
  width: 200px;
  height: 200px;
}

h1 {
  …(以下省略)…
```

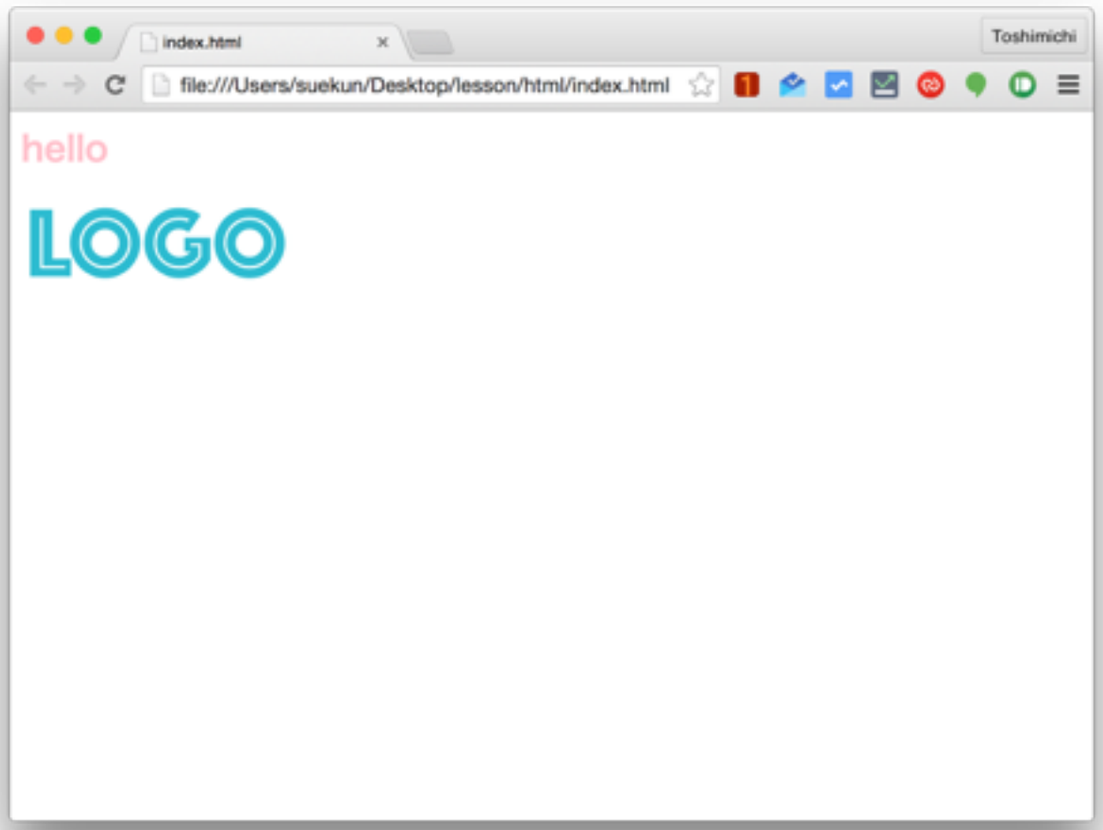
※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

index.html (./html/index.html)

```
<html>
  <head>
    <link rel="stylesheet" href="./css/style.css">
  </head>

  <body>
    <h2>hello</h2>
    <div class="logo-mark"></div>
  </body>
</html>
```



※本章の内容は旧プラグイン CSS-Sprite を紹介しておりますが、CSS-Sprite は新版として Sprity という名称で新たに公開されております。Sprity を用いた Windowsでの環境構築も含めた Retina対応のCSS スプライト自動化についての別資料を別途用意します。

4. Sassファイルの使い方

この後は position: absolute; など、
使いやすい形で利用してください

お疲れ様でした