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

追加資料

第8章　CSSスプライトを自動化しよう（sprity版）

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8章　CSSスプライトを自動化しよう（sprity版）

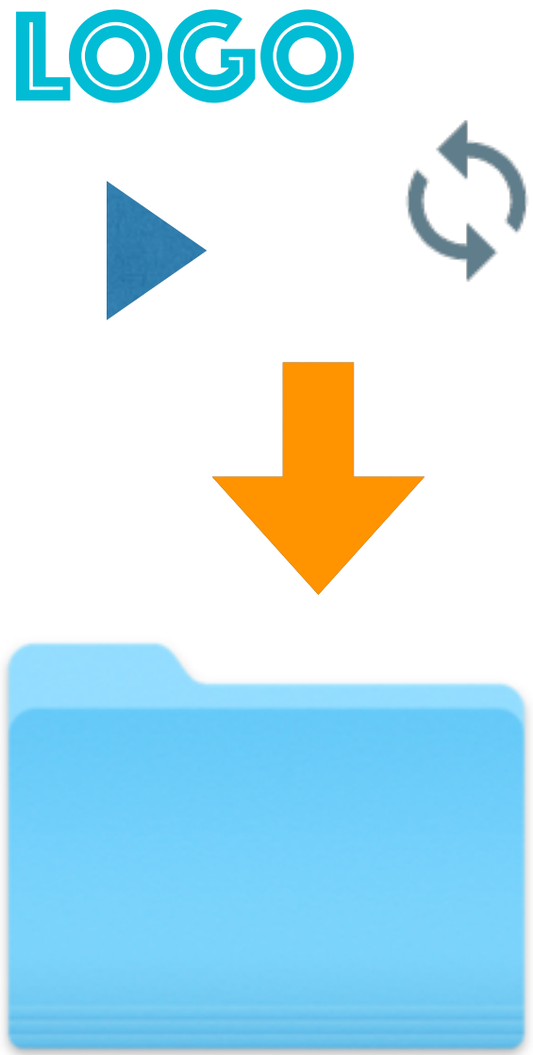
第8章　CSSスプライトを自動化しよう（sprity版）

第8章　CSSスプライトを自動化しよう（sprity版）

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

1. 概要

作業用のディレクトリに保存

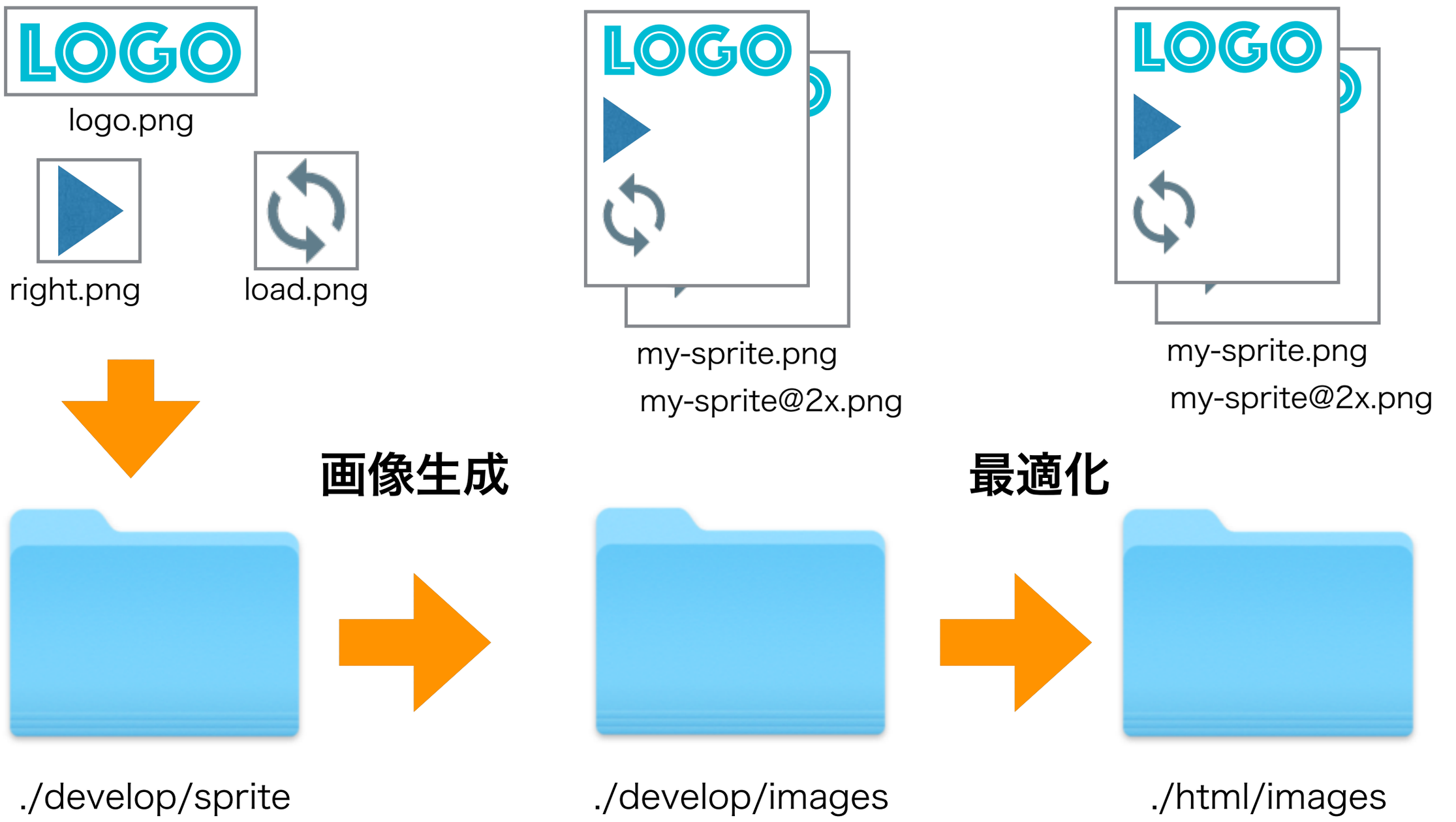


./develop/sprite

CSSスプライトが生成される

./html/images

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



2. 環境準備

2. 環境準備

Mac OSX

1. Xcode（コマンドラインツールが必要）

Windows

1. Python 2.7.x　（3.xはNG）
2. Visual Studio Community 2015
3. Visual C++ 2015 Tools for Windows Desktop

2. 環境準備 - Mac

環境構築手順（Mac）

1. Mac App Store でXcodeを入手

2. 一度起動します

3-a. GUIからコマンドラインツールをインストール

Xcode > preference… > Downloads > Command Line Tools

3-b. ターミナルからコマンドラインツールをインストール

インストール

```
$ xcode-select –install
```

インストール完了しているか確認

```
$ gcc --version
```

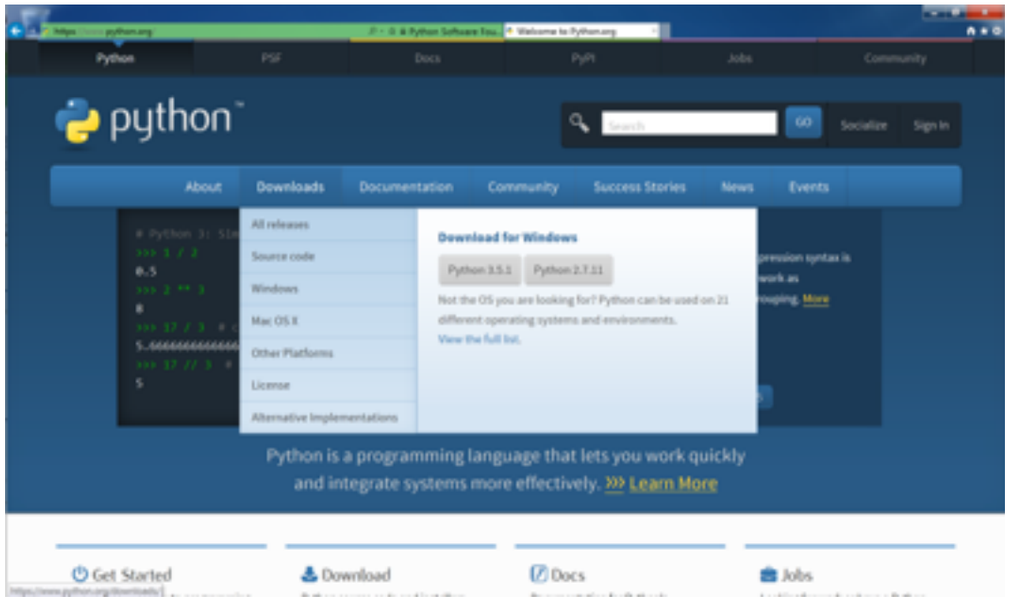
2. 環境準備 - Windows

環境構築手順（Windows）

1. Python をインストール

Python
https://www.python.org/

※インストール時には必ず[Add python.exe to Path]で[Will be installed on local hard drive]を選択しておく



コマンドプロンプトで確認

```
> python --version
```

2. 環境準備 - Windows

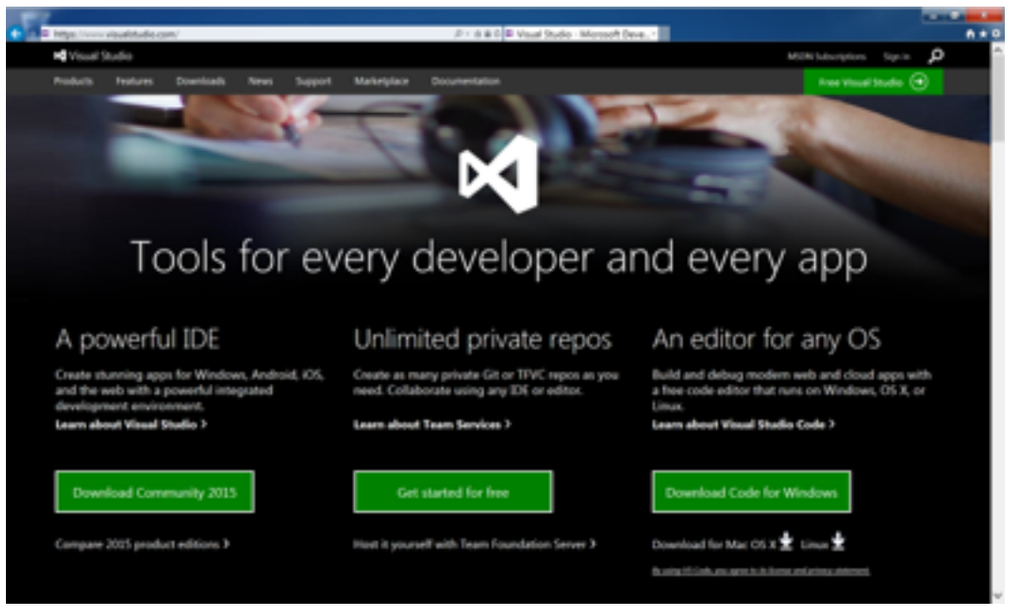
環境構築手順（Windows）

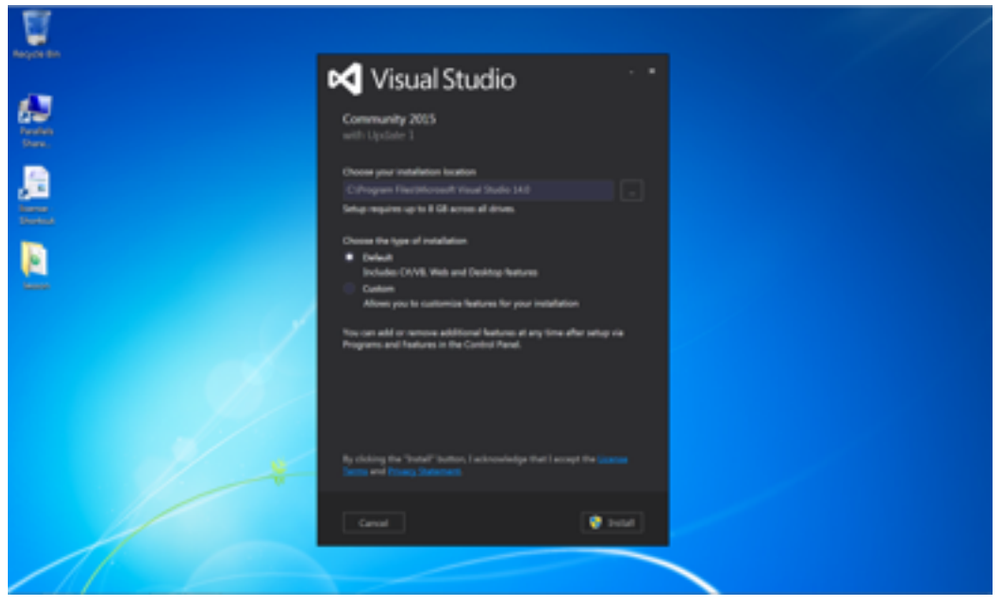
2. Visual Studio をインストール

Visual Studio Community 2015
https://www.visualstudio.com/

→ Download Community 2015

※インストールにかなり時間がかかります
※インストール時にIE10以上を要求されます
※Micrsoft account が必要です





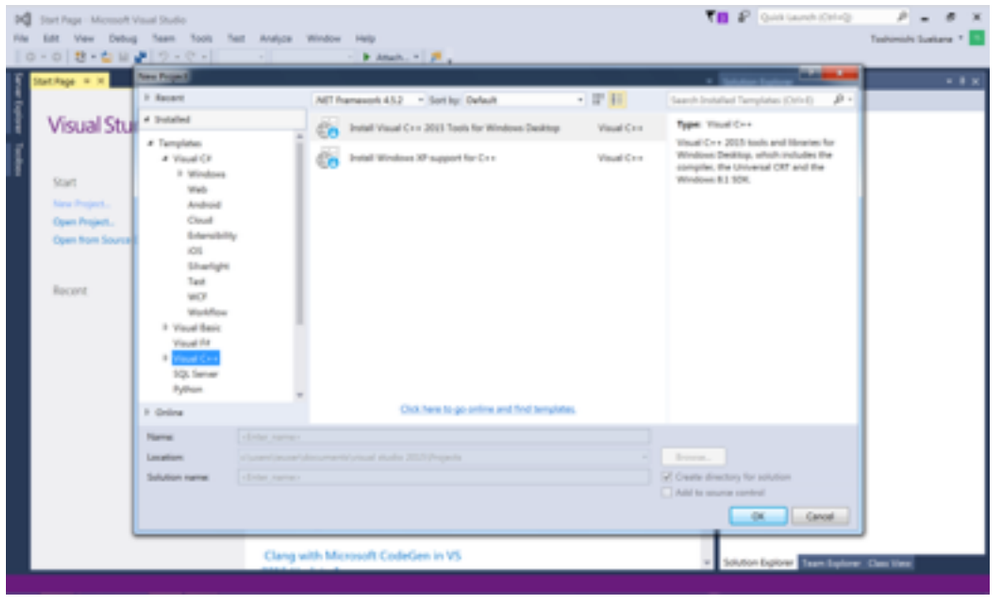
2. 環境準備 - Windows

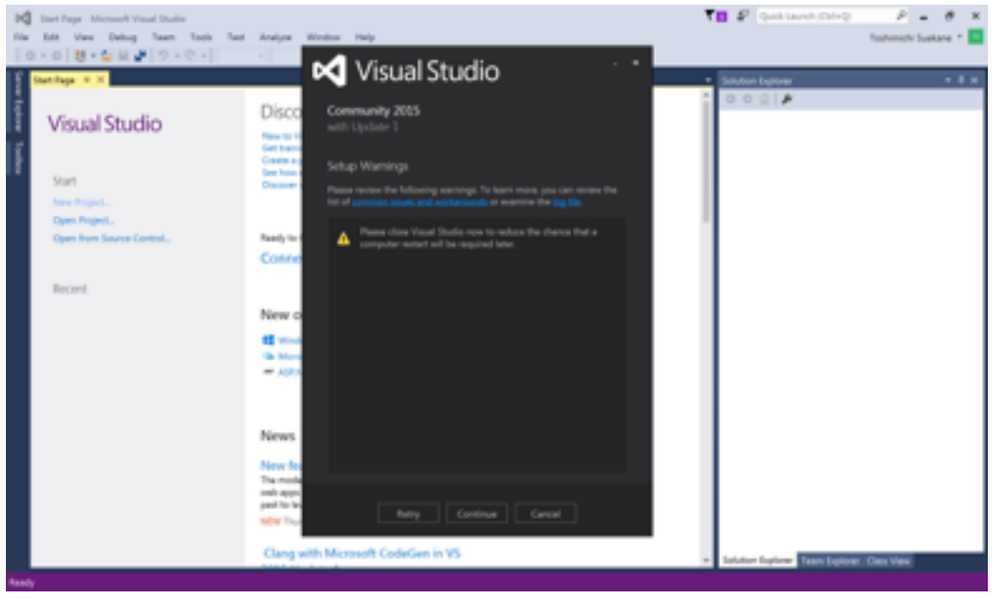
環境構築手順（Windows）

3. Visual C++ 2015 Tools をインストール

・Visual Studio Community 2015 を起動
・サイドから New Project… を選択
・Install Visual C++ 2015 Tools for Windows Desktop を選択
・OKを押してインストール実行
・Visual Studio Community 2015 を閉じるよう指示されるので終了する

※インストールにかなり時間がかかります





3. タスクを作ろう

3. 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

3. 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

3. タスクを作ろう

sprity

https://www.npmjs.com/package/sprity

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

sprity-sass

https://www.npmjs.com/package/sprity-sass

sprity 用の SASS/SCSS プロセッサー

gulp-if

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

条件分岐を使えるようになります

3. タスクを作ろう

Windows の方

css-sprite を追加

```
> npm install sprity –save-dev -g
```

sprity-sass を追加

```
> npm install sprity-sass --save-dev
```

gulp-if を追加

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

Mac の方

css-sprite を追加

```
$ npm install sprity –save-dev -g
```

sprity-sass を追加

```
$ npm install sprity-sass --save-dev
```

gulp-if を追加

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

3. 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

3. タスクを作ろう

Mac の方

sprite ディレクトリを作成

```
> mkdir develop\sprite
```

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

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

gulp_task 内に sprite.js ファイルを作成

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

3. 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
```

3. 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/*.png', // 監視・対象ファイル
    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: 'sprity-sass',        // sprity-sassを使用
    margin:    0,                    // 画像間のマージン
    type:      'sass',               // Sassの種類
  },
};
```

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

3. タスクを作ろう

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

```
var gulp   = require('gulp');
var gulpif = require('gulp-if');
var sprity = require('sprity');
var conf   = require('../config');

gulp.task('sprite', function () {
  return sprity.src({
    src:           conf.sprite.src,
    name:          conf.sprite.name,
    style:         conf.sprite.style,
    cssPath:       conf.sprite.cssPath,
    processor:     conf.sprite.processor,
    prefix:        conf.sprite.prefix,
    margin:        conf.sprite.margin,
    'style-type': conf.sprite.type,
    'dimension':  [{
      ratio: 1, dpi: 72
    }, {
      ratio: 2, dpi: 192
    }],
  })
  .pipe(
    gulpif('*.png',
      gulp.dest(conf.sprite.dest),
      gulp.dest(conf.sprite.sass)
    )
  )
});
```

sprityの公式サイトでは分かりにくい部分があるので、
あらかじめサンプルを用意しました

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

3. 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']);
  });
});
```

spriteタスクで生成したのち、imageタスクで最適化

```
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

4. 動作確認をしよう

4. 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

4. 動作確認をしよう

develop/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
```

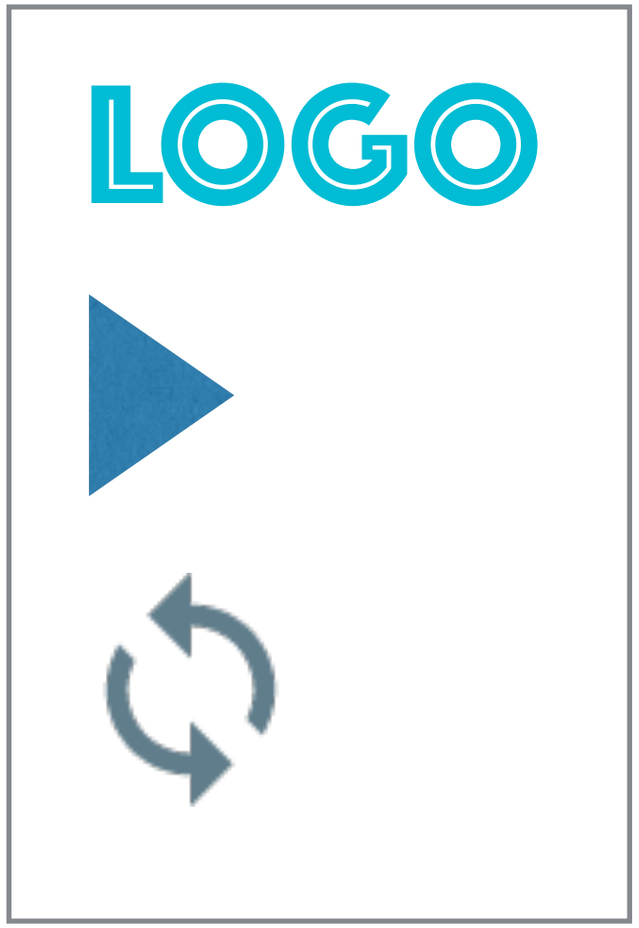
意図した通り動作していればOKです

5. Sassファイルの使い方

5. Sassファイルの使い方

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

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

5. Sassファイルの使い方

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

```
@import sprite

.logo-mark
  +sprite($logo)
```

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

5. Sassファイルの使い方

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

```
@import sprite

.logo-mark
  +sprite($logo)
```

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;
}
```

5. Sassファイルの使い方

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

```
@import sprite

.logo-mark
  +sprite($logo)
```

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;
}
```

この状態では background-image の指定が効いていません

5. Sassファイルの使い方

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

```
@import sprite

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

生成されれいる .sprite クラスを読み込みます

5. Sassファイルの使い方

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

```
@import sprite

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

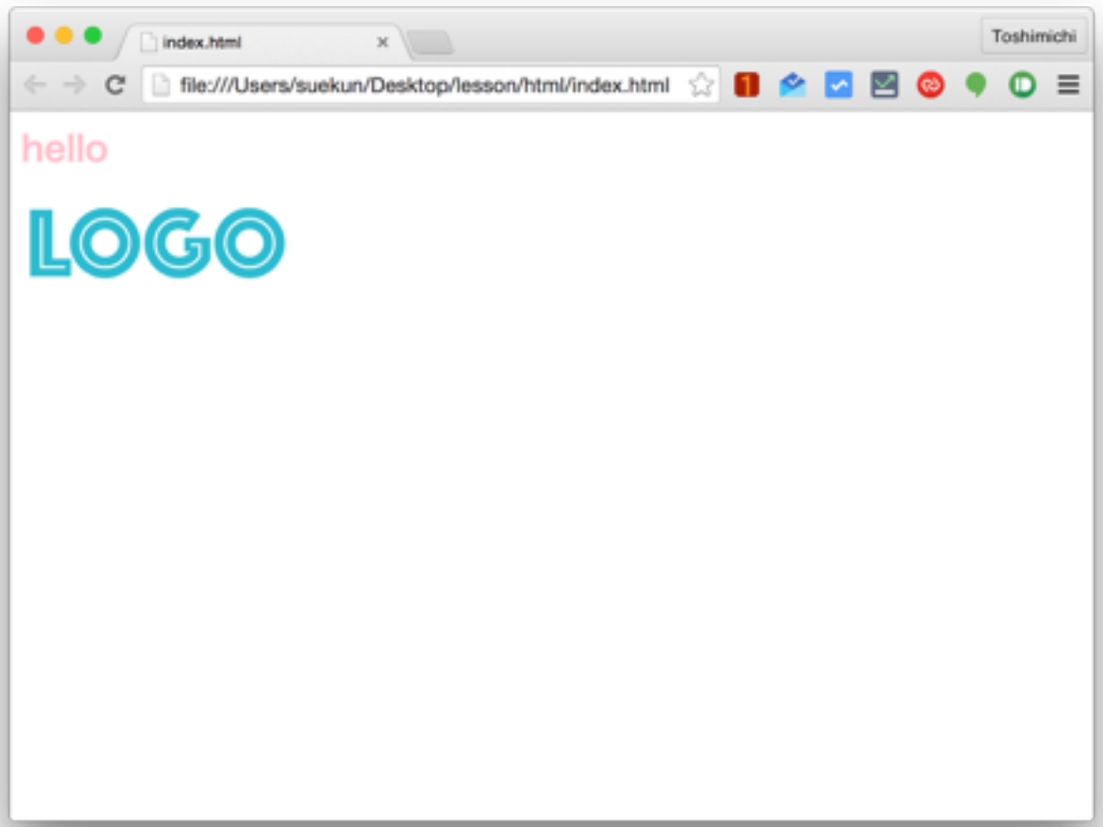
style.css (./home/css/style.sass)

```
.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;
}
```

5. 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>
```



5. 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