Gulpを使用する

前回の環境構築でgulpの導入までは終わりました。

ただまだ何もタスクを作成してないので何もできない状態です。

実際にタスクを作成し、実行してみましょう！

タスクの作成

まずは下記のコマンドでgulpの設定ファイルを作っていきます。

`touch gulpfile.js`

できた下記のコードをコピペしてください。

```
// gulpプラグインの読み込み
const gulp = require("gulp");
// Sassをコンパイルするプラグインの読み込み
const sass = require("gulp-sass");

// style.scssをタスクを作成する
gulp.task("default", function() {
  // style.scssファイルを取得
  return (
    gulp
      .src("sass/style.scss")
      // Sassのコンパイルを実行
      .pipe(sass())
      // cssフォルダー以下に保存
      .pipe(gulp.dest("css"))
  );
});
```

2、4行目に`require('プラグイン名')`で使用するプラグインを読み込んでいます。gulp関連のプラグインを追加した時は必ず読み込むようにしてください。

`gulp.task('タスク名', 実行される処理)`タスク名と実行する処理を書いていきます。

タスク名を`default`としておくと実行時にタスク名を書かずとも実行してくれます。

実行するタスクは`return`で囲むようにしましょう。

`gulp.src()`でタスクの対象となるファイルを取得します。複数のファイルも指定できます。

複数してする場合は`*.scss`とすれば対象のディレクトリのすべてのSassを見てくれます。

また、`sass/**/*.scss`のようにパスを指定するとsassディレクトリ直下のすべてのディレクトリを見ます。

`pipe()`は1つ1つの処理をつなげます。今回だと`src()`で書いたパスに対して、`sass()`でコンパイルを実行し、コンパイルでできあがったものを`gulp.dest("css")`でCSSディレクトリ名を指定して出力しています。もしディレクトリが作られてなくても自動で作ってくれます。

サンプルでそうしていますが、`pipe()`はいくらでもつなげることができます。

別々で書くことも可能なのでしょうがそうすると、1つ1つの処理のコードが長くなる、共有した時にどんな処理をするのか把握するのが難しくなります。

`src()`指定したディレクトリ名、ファイル名がまだできていないので作成し、作成したSassファイルの中に下記のスタイルを書いてください。

```
.main {
  font-size: 20px;
  &-content {
    font-size: 14px;
  }
}
```

それができたら下記のコマンドを実行してください。

```
npx gulp
```

実行後`gulp.dest()`で書いたディレクト名が作成され、その中にCSSファイルが作成されていたらその中を見てください。

Sassで書いた内容と同じ内容のCSSができていたら成功です。

開いて見るとちょっといつも自分で書く時とちょっと違いますよね？

いつものような感じにしたい場合は13行目の`sass()`の中に`{outputStyle: "expanded"}`入れて再度タスクを実行して見てください。

でき上がったものを開くと見慣れたCSSになっています。

逆にCSSはコンピューターしか見ない。スタイルはSass見ればわかると言った場合は1行にまとめる方法もあります。

どんなことができるかは調べてみてください。

watch機能

開発中Sassファイルは頻繁に更新されます。その都度コマンド実行は少々面倒です。

watch機能を使用することで変更を検知し自動的にCSSにコンパイルしてくれます。

下記のコードを先ほど作ったgulpファイルにコピペして上書きしてください。

```
// gulpプラグインの読み込み
const gulp = require('gulp');
// Sassをコンパイルするプラグインの読み込み
const sass = require('gulp-sass');

// style.scssの監視タスクを作成する
gulp.task('default', function () {
  // ★ style.scssファイルを監視
  return gulp.watch('sass/style.scss', function () {
    // style.scssの更新があった場合の処理

  // style.scssファイルを取得
  return gulp.src('sass/style.scss')
    // Sassのコンパイルを実行
    .pipe(sass({
      outputStyle: 'expanded'
    })
      // Sassのコンパイルエラーを表示
      //（これがないと自動的に止まってしまう）
      .on('error', sass.logError))
    // cssフォルダー以下に保存
    .pipe(gulp.dest('css'));
  });
});
```

先ほどと変わった部分は9行目に`watch()`と20行目の`op()`が加わりました。

watch機能は`watch('監視対象'、'実行するタスク')`で使用することができ、今回はSassファイルが更新sれた時に`function () {}`の中の処理を実行するように書いています。

`.on('error', sass.logError)`はコンパイル時にエラーが発生した時にそのエラー内容をターミナルに表示します。

この時はだいたいSassの書き方に問題がありますが、エラー内容見て対応しましょう。

再度`npx gulp`を実行しSassファイルを更新してみてください。

更新した内容が自動的にCSSファイルへ反映されたいたら成功です。

この他にも様ざなタスクがあります。

すべてを紹介はできないので自分で調べ実際に実装してみてください・

課題

以下の項目を調べてClassroomに提出してください。

- `require()`どう言った時に使いますか？
- `gulp.task()`どう使いますが？また、どう言った仕様ですか？
- タスクの対象はどのようにして指定すれば良いですか？
- 対象のタスクに対して複数処理をしたい時はどうすれば良いですか？
- watch機能はどう言ったものですか？