Les Task Runner

Qu’est ce qu’un Task Runner ?

Un Task Runner permet tout simplement d’automatiser les taches régulières qui font perdre du temps.

Lors de votre projet web, vous faites des retours clients pour montrer son état d’avancement. Pour cela, vous devez **ajouter des préfixes** pour gérer l’affichage dans différentes versions de navigateur. Vous devez aussi **minifier** votre CSS et votre Javascript ainsi que compresser des images...

Il existe plusieurs Task Runner, comme Gulp nous allons voir mais il en existe d’autre Grunt, Brunch.

Installation de Gulp

Il faut installer une interface de ligne de commande pour Gulp, comme ceci :

```
npm install --global gulp-cli
```

Puis vous installez Gulp dans votre projet comme ceci :

```
npm install --save-dev gulp
```

Gulp à présent installé, il lui faudra une combinaison de deux fichiers à la racine d'un projet pour être totalement fonctionnel :

- **`package.json :`** contient la liste des plugins gulp (ou autres) nécessaires à vos tâches
- **`gulpfile.js :`** contient la liste des tâches à réaliser

Le fichier gulpfile.js

Le fichier gulpfile.js s'occupe de gérer les tâches à réaliser, leurs options, leurs sources et destination. Bref, c'est le chef d'orchestre.

Créez le fichier gulpfile.js à la racine de votre projet.

Dans le fichier gulpfile.js nous allons importer les fonctions de gulp dans une variable comme ceci :

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

Une tâche gulp s'écrie comme ceci :

```
gulp.task('nom_de_ma_tache', function () {
    // Les pluggins devant s'exécuter lorsque ma tâche est appelée
});
```

Pour l'éxécuter il faudra taper dans votre console : `gulp nom_de_ma_tache`

Créez votre première tâche, comme ceci :

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

Cette tâche affiche "`Hello world !!!`" dans la console (terminal).

Pour l'exécuter taper :

```
gulp hello
```

Vous savez maintenant créer une tâche gulp, nous allons pouvoir ajouter notre première tâche qui gérera le Sass.

Gulp-sass

Avant toute choses notre workflow va générer des fichiers, puis les transformer, etc... Il est donc important que nous réorganisions notre espace de travail.

Nous allons créer un dossiers "src" pour source, ce sera notre dossier de travail.

Dans ce dossier “src” ajoutons un fichier index.html et un dossier assets. Dans ce dossier assets créons un dossier “sass” et “css” notre espace de travail devrai ressembler à ceci :

Maintenant installons les modules qui nous serons utiles afin de compiler notre Sass en Css comme ceci :

```
npm install sass gulp-sass --save-dev
```

Ajoutons en haut de notre fichier gulpfile.js, à la suite de notre première variable “gulp”, ceci :

```
const sass = require('gulp-sass')(require('sass'))
```

Ajoutons maintenant une tâche qui gérera la compilation du Sass.

```
gulp.task('sass', function () {
  return gulp.src('./src/assets/sass/**/*.scss')
    .pipe(sass())
    .pipe(gulp.dest('./src/assets/css'));
})
```

Le gulp.task() crée une nouvelle tâche, que nous appelons ‘sass’. Le gulp.src() crée un flux pour lire tous les fichiers SCSS. Avec pipe() nous transmettons les données diffusées au sass compilateur. Enfin, nous transmettons les données compilées au gulp.dest(), qui à son tour crée un flux pour écrire les données dans le système de fichiers.

Cette tâche ne gère que la compilation du sass en css, il ne gère pas la surveillance des modification (watch).

Pour exécuter cette tâche il suffit simplement de taper dans votre terminal et de d’appuyer sur la touche ENTRER :

```
gulp sass
```

Pour surveiller automatiquement les changements, Gulp dispose d'une fonction watch().

Pour cela nous allons créer une nouvelle tâche, que nous allons nommer 'sass:watch' qui va permettre de surveiller la modification de fichier scss, si c'est le cas cette nouvelle tâche va permettre l'exécution de la tâche 'sass' crée précédement.

C'est parti :

```
gulp.task('sass:watch', function () {
  gulp.watch('./src/assets/sass/**/*.scss', gulp.series('sass'));
});
```

La fonction series() combine les fonctions de tâche et / ou les opérations composées en opérations qui seront exécutées les unes après les autres, dans un ordre séquentiel.

Il existe une autre fonction parallel() qui permet d'exécuter les fonctions et / ou opérations qui seront exécuter en parallèles.