Installation de Sass sur Linux

Pour installer `Sass` nous allons utiliser `npm`, et pour utiliser `npm`, il faut installer `Node.js`.

Nous allons donc voir très brièvement ce que sont `npm` et `Node.js`, mais ne vous en faîtes pas, dans ce chapitre vous n'aurez besoin que de connaître que très peu de choses à leur sujet.

Vous les reverrez dans les cours plus avancés très en détails.

Qu'est-ce que `npm` ?

`Npm` veut dire `Node Package Manager` et couvre en fait **trois choses différentes**.

Premièrement, c'est **le gestionnaire de paquets officiel de** `Node.js`. Il est installé par défaut lorsque vous installez `Node.js`. **Il est aujourd'hui plus globalement le gestionnaire de paquets le plus utilisé pour le JavaScript**. `Node.js` est un environnement serveur permettant d'exécuter du `JavaScript` côté serveur.

Deuxièmement, c'est le plus important dépôt de paquets (*packages registry*) au monde et contient à ce jour plus de 900000 paquets `JavaScript` !

Troisièmement, c'est un CLI, c'est-à-dire une interface en ligne de commande, permettant de gérer les paquets dans un projet : les installer, les désinstaller, les mettre à jour etc. Il permet également de publier des paquets sur le dépôt `npm`.

Installation de `Node.js`

Pour installer `Node.js` quel que soit votre OS c'est très simple.

Vous pouvez utiliser un exécutable en le téléchargeant ici.

Ou vous pouvez utiliser un gestionnaire de paquets ici.

Si vous êtes sur `Ubuntu`, `Debian`, `redhat`, `centOS` ou `fedora` il suffit de vous rendre ici et de choisir l'avant dernière version de `Node` qui est la `LTS`.

Que ce soit sur `Linux` ou sur `MacOS`, ouvrez rechercher et tapez simplement `terminal`.

Vous pouvez ensuite ouvrir votre terminal et faire : `node -v`.

Vous pouvez également ouvrir un terminal dans `VS Code`.

Installation de `git`

Par défaut, sur `Windows`, le terminal est `Powershell`.

Dans tous les cours nous utilisons `bash`, qui est le terminal le plus utilisé et que vous retrouverez sur les serveurs et sur la plupart des environnements de développement.

Nous allons donc installer `git` pour pouvoir l'utiliser : téléchargez le ici.

Utilisation de `Git Bash` sur `Windows`

Sur `Windows` uniquement, ouvrez `VS Code`.

Faites `Ctrl + Shift + p` ou `View` puis `Command Palette`.

Entrez `select default shell` puis faites entrée.

Ensuite sélectionnez `Git Bash`.

Vous pouvez ensuite faire `Terminal` puis `New Terminal` et vous aurez un terminal `Bash` !

Créer un fichier `package.json`

Créez un nouveau dossier, puis faites `npm init` dans le dossier. Cela créera un fichier `package.json`.

Le fichier `package.json` permet de :

Lister les paquets utilisés par votre projet qui sont appelés ses dépendances. En effet, sans ces paquets, votre projet ne fonctionne pas, il est donc bien dépendant de celles-ci.

Spécifier précisément quelles versions des dépendances peuvent être installées.

Permettre aux développeurs d'installer toutes les dépendances de votre projet en une commande.

Installation de votre première dépendance : `sass`

Pour installer une nouvelle dépendance dans un projet qui a un `package.jon`, il suffit d'utiliser `npm` et de faire :

```
npm install nom
```

Ou en utilisant le raccourci `i` :

```
npm i nom
```

Ces paquets seront installés dans le dossier `node_modules` et seront ajoutés comme dépendance dans le fichier `package.json` :

```
{
  "name": "my_package",
  "version": "1.0.0",
  "dependencies": {
    "ma_dependance": "^1.0.0",
  }
}
```

Pour installer `sass` il suffit donc de faire :

```
npm i sass
```

Le fichier `package-lock.json`

Vous remarquerez que lorsque vous installez une dépendance avec `npm i` un fichier `package-lock.json` est créé.

Retenez seulement que les algorithmes de gestion des dépendances sont très complexes et que le même fichier `package.json` peut aboutir à des installations différentes.

Pour s'assurer que l'arbre des dépendances (vos dépendances ont elles-mêmes des dépendances résultant en un arbre immense de dépendances) installé soit le même entre deux `npm i` avec le même `package.json`, `npm` crée un fichier `package-lock.json` qui est une image figée de l'arbre de toutes les dépendances (et des dépendances de vos dépendances etc), de votre projet.

Premier `script`

Dans notre fichier `package.json`, vous remarquez qu'il y a une entrée `scripts`.

Ces `scripts` peuvent être lancés en faisant `npm run nom-du-script`.

Nous allons créer notre premier `script` :

```
{
  "name": "sass",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "sass": "sass -w scss:css"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "sass": "^1.32.11"
  }
}
```

`sass` permet de lancer la librairie `sass`.

`--watch` ou `-w` : permet d'indiquer le dossier ou les fichiers contenant les fichiers `Sass` qui vont être suivis et converti en `CSS` automatiquement.

`--output` ou `-o` : permet d'indiquer le dossier où seront mis les fichiers `CSS` convertis depuis le `Sass`.

Nous pouvons ensuite lancer le `script` en faisant `npm run sass`.

Si vous exécutez la commande dans le terminal, vous verrez qu'il reste bloqué. C'est tout à fait normal car il est en mode `watch` : il attend que des fichiers soient modifiés dans le dossier `sass` pour les compiler à chaque fois.

Pour couper l'exécution vous pouvez faire `Ctrl + c`.

Création de dossiers et fichiers pour tester

Dans le dossier, créez un dossier `scss` pour les fichiers `Sass` et un dossier `css` pour les fichiers `css`.

Créez un fichier `index.html` et placez-y le template `!` :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
```

```
1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <title>Sass/title>
  </head>
  <body>
  </body>
</html>
```

Ensuite, dans le dossier `scss` créez un fichier `style.scss` et nous allons écrire notre premier code `Sass` :

```
body {
  background: red;
  height: 100vh;
}
```

Vous pouvez ensuite lancer le `script` avec `npm run sass` puis sauvegarder le fichier.

Vous pourrez observer qu'un fichier a été créé automatiquement en `CSS` dans le dossier `c ss` !

Remarquez déjà deux choses : l'extension des fichiers `Sass` est `.scss` et le code `CSS` fonctionne en `Sass` .