Les bon outils CSS font les bons ouvriers du Web

Alvin Berthelot

www.webyousoon.com

Alvin Berthelot

Email : alvin.berthelot@webyousoon.com

Twitter : @alvinberthelot

Une application Web aujourd'hui

Utilisable et maintenable

Évolutive

Compatible

Rapide

Fiable

CSS dans tout ça ?

CSS est **le langage** de mise en forme du Web

Il faut donc être conscient que **les exigences d'une application doivent s'appliquer également à la production de feuilles de styles**

De **nombreux outils** existent pour nous aider à atteindre ces exigences

Un outillage adapté

Maintenable

Générer un guide de style

EXAMPLE


```
<span class="label label-default">Default</span>
<span class="label label-primary">Primary</span>
<span class="label label-success">Success</span>
<span class="label label-info">Info</span>
<span class="label label-warning">Warning</span>
<span class="label label-danger">Danger</span>
```

Copy

Comprendre rapidement ce qui existe et peut-être réutilisé

Vecteur de communication

Assurer une cohérence code et rendu

Évolutive

Utiliser un préprocesseur CSS

- Il s'agit d'un **métalangage**, ne pouvant être utilisé comme tel, mais servant à faciliter la génération d'un autre langage (en l'occurence CSS)

- C'est un **outil** pour produire et maintenir des feuilles de styles CSS, cela ne vient pas remplacer le standard qu'est CSS pour la mise en forme

- L'utilisation d'un préprocesseur CSS ne vous offre donc **pas de nouvelles possibilités** sur la mise en forme

Pourquoi un préprocesseur CSS ?

Si CSS reste le maître incontesté de la mise en forme, son utilisation brute est toutefois perfectible sur certains points :

- Le **découpage des fichiers sources**
- Les **dépendances de mises en forme** pour garder une cohérence globale
- La **répétitivité de certaines tâches**

Quels sont les préprocesseurs existants ?

Il existe plusieurs préprocesseurs CSS, mais à l'heure actuelle, 3 ressortent clairement du panier :

- **Sass** : Préprocesseur écrit en Ruby mais désormais également écrit en C (LibSass) ce qui a permis le portage sous Node.js
- **LESS** : Préprocesseur écrit en ECMAScript s'executant sous Node.js
- **Stylus** : Préprocesseur écrit en ECMAScript s'executant sous Node.js

Les 3 préprocesseurs ont gloablement des fonctionnalités similaires, seules leurs syntaxes sont différentes.

Les fonctionnalités

L'écriture de mise en forme CSS directe (sans préprocesseur) étant perfectible, quelles sont les fonctionnalités qu'offre Sass pour palier à ces problèmes :

- **Import** : inclusion de fichiers Sass
- **Variable** : utilisation de variables
- **Nesting** : imbrication des sélecteurs CSS
- **Operations** : réalisation d'opérations sur les valeurs
- **Function** : utilisation de fonctions pour calcul d'une valeur
- **Mixin** : composition de propriétés CSS
- **Extend** & **Placeholders** : héritage de propriétés CSS

Import (exemple)

Sass

```
body {
  background: whitesmoke;
}

// Fichier contenant h1 {color: blue;}
@import "partials/typo";
```

CSS

```
body {
  background: whitesmoke;
}

h1 {
  color: blue;
}
```

Variable (exemple)

Sass

```sass
// Déclaration d'une variable avec $
$purple: #9013FE;

h1 {
  // Utilisation
  color: $purple;
}
```

CSS

```css
h1 {
  color: #9013FE;
}
```

Nesting (exemple)

Sass

```
section {
  border: 1px solid whitesmoke;
  .alert {
    colour: red;
  }
}
```

CSS

```
section {
  border: 1px solid whitesmoke;
}
section .alert {
  colour: red;
}
```

Opérations (exemple)

Sass

```
$container-width: 960px;
$num-columns: 3;

.column {
  width: container-width / num-columns;
}
```

CSS

```
.column {
  width: 320px;
}
```

Function (exemple)

Sass

```sass
@function pxtorem($pxval, $base) {
  @return ($pxval / $base) * 1rem;
}

h1 {
  margin: pxtorem(24px, 16px);
}
```

CSS

```css
h1 {
  margin: 1.5rem;
}
```

Les fonctions natives Sass (exemple)

Sass

```
$colour: red;
a {
  colour: darken($colour, 10);
}
a:hover {
  colour: lighten($colour, 10);
}
```

CSS

```
a {
  colour: #cc0000;
}
a:hover {
  colour: #ff3333;
}
```

Mixin (exemple)

Sass

```sass
@mixin text-truncate {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}
.teaser {
  width: 300px;
  @include text-truncate;
}
```

CSS

```css
.teaser {
  width: 300px;
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}
```

Extend (exemple)

Sass

```
.info {
  border-radius: 4px;
  padding: 20px;
}
.alert {
  @extend .info;
  background: red;
}
```

CSS

```
.info, .alert {
  border-radius: 4px;
  padding: 20px;
}
.alert {
  background: red;
}
```

Placeholders (exemple)

Sass

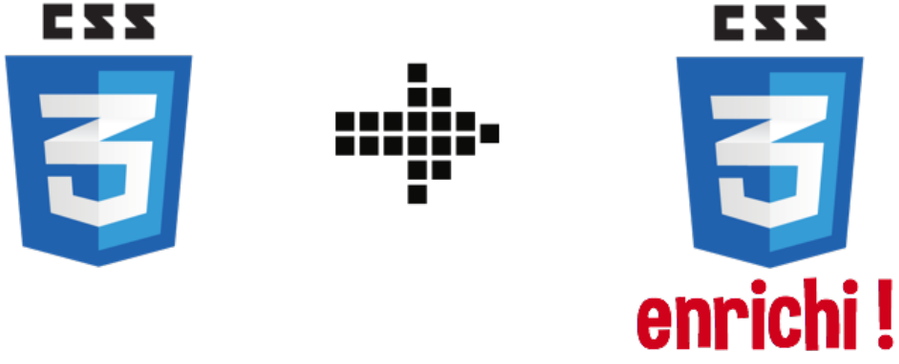
```
%info {
  border-radius: 4px;
  padding: 20px;
}
.alert {
  @extend %info;
  background: red;
}
```

CSS

```
.alert {
  border-radius: 4px;
  padding: 20px;
}
.alert {
  background: red;
}
```

Compatible

Utiliser un post-processeur CSS



- Contrairement au préprocesseur, ce n'est pas un **métalangage**
- C'est un **outil** pour enrichir et maintenir des feuilles de styles CSS

Post-processeur (exemple)

CSS

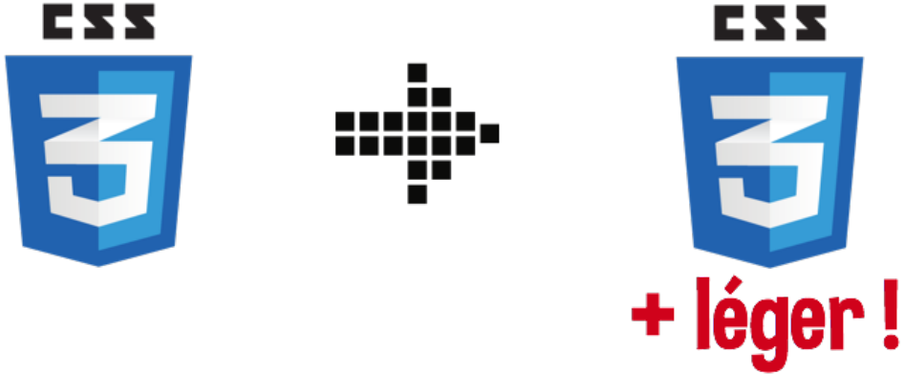
```css
a {
  transition: transform 1s;
}
```

CSS

```css
a {
  -webkit-transition: -webkit-transform 1s;
  transition: -ms-transform 1s;
  transition: transform 1s;
}
```

Rapide

Minifier



- Réduit le poids d'une feuille de styles CSS en ne gardant que ce dont le navigateur à réellement besoin

- Suppression des commentaires, des espaces, des sauts de lignes, etc.

- Mais cela peut aller bien plus loin (regroupement de propriétés, de media queries, etc.) et tous les "minificateurs" n'ont pas les mêmes performances

Épurer le CSS

Êtes vous certain d'utiliser toutes les propriétés CSS que vous avez déclaré ?

Pour le savoir c'est simple comparons celles déclarées avec celles utilisées, il y a un **uncss** pour ça

Fiable

Utiliser des tests de rendu

C'est possible, mais c'est loin d'être si simple … Restez pragmatique

Conclusion

Vous devriez vraiment utiliser Node.js pour la richesse de son écosystème

Globalement, la mise en place d'outils CSS est assez simple et peut vraiment vous faire gagner en qualité et performance

Les pré-processeurs peuvent complexifier certaines choses, mais vous devriez quand même regarder car toutes les grosses librairies CSS sont basées dessus

Tous ces outils ne vous abstiennent pas de **connaître CSS**