A BOOK APART
Les livres de ceux qui font le web

No. 10

Dan Cederholm

SASS POUR LES WEB DESIGNERS

PRÉFACE DE Chris Coyier

EYROLLES

1 POURQUOI SASS ?

J'AI EU DU MAL à croire en Sass. J'écris mes feuilles de styles à la main ! Je n'ai pas besoin d'aide ! Et je n'ai pas envie qu'on complique mon workflow. Fichez-moi la paix !

C'est ce que je me disais, en tout cas. Mais la vérité, c'est que Sass (et d'autres préprocesseurs CSS) peut être un allié puissant – un outil que n'importe quel web designer saura incorporer facilement à son travail quotidien. J'ai mis du temps à me faire à cette idée, mais je ne regrette pas d'avoir franchi le pas.

C'est pour cette raison que j'ai voulu écrire ce petit livre. Pour vous faire part de mon expérience avec Sass, qui m'a permis de devenir plus efficace tout en préservant la méthode que j'utilise depuis dix ans pour composer mes feuilles de styles. J'avais de nombreux préjugés sur Sass qui m'empêchaient de me lancer. Je craignais d'avoir à changer complètement ma façon d'écrire et de gérer mes feuilles de styles. CSS peut parfois être fragile, et il est compréhensible que ses utilisateurs soient un peu sur la défensive. Pas vrai ?

Je suis donc ici pour vous démontrer que Sass ne bouleversera pas votre méthode de travail, et qu'il vous rendra même la vie

plus facile. Je détaillerai les divers aspects de Sass, son installation, son utilisation et l'aide qu'il m'a apportée dans mes propres projets. Avec un peu de chance, je parviendrai à vous convertir.

SASS EN QUELQUES MOTS

Il vous est déjà arrivé de vouloir changer une couleur dans votre feuille de styles et de vous apercevoir que vous deviez la remplacer à plusieurs endroits n'est-ce pas ? Vous n'aimeriez pas que CSS vous permette de faire ça plus facilement ?

```
$brand-color: #fc3;

a {
  color: $brand-color;
}
nav {
  background-color: $brand-color;
}
```

Avec Sass, vous pouvez changer cette valeur à un seul endroit et la propager dans toute la feuille de styles. C'est aussi simple que ça !

Qu'en est-il des blocs de style que vous réutilisez plusieurs fois au long de votre feuille de styles ?

```
p {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}
footer {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}
```

Ne serait-il pas fantastique de pouvoir regrouper toutes ces règles partagées dans un bloc réutilisable ? Là encore, Sass vous

permet de les définir une seule fois et de les inclure quand vous en avez besoin.

```
@mixin default-type {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}

p {
  @include default-type;
}
footer {
  @include default-type;
}
```

Sass, c'est aussi ça ! Et ces deux exemples extrêmement simples ne suffisent pas à décrire les gains de temps et de flexibilité potentiels. Sass est une aide bienvenue dans le monde du web design, car quiconque a déjà créé un site web sait que...

CSS, C'EST COMPLIQUÉ

Soyons réalistes : apprendre le CSS, ce n'est pas si simple. Comprendre la fonction de chaque propriété, le principe de la cascade CSS, quel navigateur prend en charge quoi, les sélecteurs et autres bizarreries. Ajoutez à cela la complexité des interfaces que nous construisons de nos jours, et la maintenance qui va avec et... Attendez, pourquoi on fait tout ça déjà ? C'est un puzzle, et certains d'entre nous aiment en assembler toutes les pièces.

Une partie du problème vient du fait que CSS n'a pas été conçu à la base pour tout ce qu'on veut lui faire faire aujourd'hui. Bien sûr, les normes progressent grâce à l'évolution rapide des navigateurs et à l'implémentation de CSS3, entre autres. Mais nous sommes toujours amenés à utiliser des techniques qui sont pour ainsi dire des bidouillages. La propriété `float`, par exemple, a été conçue à l'origine pour aligner une image dans un bloc de texte.

C'est tout. Et pourtant, nous avons tordu cette propriété dans tous les sens pour mettre en page des interfaces entières.

Par ailleurs, nos feuilles de styles sont excessivement répétitives. Pensez aux couleurs, aux polices de caractères, aux groupes de propriétés souvent réutilisés, etc. Un fichier CSS typique est un document extrêmement linéaire – du genre qui ferait s'arracher les cheveux à un programmeur orienté objet. (Je ne fais pas de programmation orientée objet, mais il me reste de toute façon bien peu de cheveux.)

À mesure que les interfaces et les applications web se font de plus en plus robustes et complexes, nous détournons la fonction originelle de CSS pour la plier à notre volonté, rusés que nous sommes. Par chance, les créateurs de navigateurs adoptent les nouvelles fonctionnalités CSS plus rapidement de nos jours, et notamment des propriétés et des sélecteurs plus efficaces et plus puissants, qui répondent aux problèmes actuels du Web. On pense par exemple aux nouvelles options de mise en page de CSS3, telles que `border-radius`, `box-shadow`, les sélecteurs avancés, les transitions, les transformations, les animations, etc. Les temps sont prometteurs. Et pourtant, CSS compte encore bien des lacunes.

Le principe DRY

Si l'on s'intéresse un instant (mieux vaut ne pas trop s'y attarder) au monde du génie logiciel, on comprend rapidement à quel point l'organisation, les variables, les constantes, les classes partielles, etc. sont essentielles et profondément ancrées dans le travail des programmeurs qui conçoivent des systèmes complexes.

Vous avez peut-être déjà entendu parler du principe de non-redondance DRY, pour « don't repeat yourself ». Défini par Andy Hunt et Dave Thomas dans leur livre *The Pragmatic Programmer* (http://bkaprt.com/sass/1/), le principe DRY énonce :

> *Dans un système, toute connaissance doit avoir une représentation unique, non ambiguë, faisant autorité.*

L'idée, c'est que la duplication de code peut être source d'erreurs et de confusion pour les développeurs (http://bkaprt.com/sass/2/). C'est également une question de bon sens : si des motifs se répètent, il paraît logique de les écrire une seule fois et

de les réutiliser à travers toute l'application. Cela donne un code plus efficace et beaucoup plus simple à entretenir.

Le CSS est tout sauf un langage DRY. Parfois, il regorge de règles, de déclarations et de valeurs redondantes. Nous réécrivons constamment les mêmes bouts de code pour définir des couleurs, des polices de caractères et des blocs de style à travers nos feuilles de styles. En parcourant un fichier CSS de bonne taille, un développeur de logiciels DRY sera d'abord muet de stupéfaction, avant de sangloter de frustration.

> « Et comment je suis censé maintenir cette !#$-@ ? vous demandera-t-il.
> – À ce propos, je t'ai parlé des bugs sous Internet Explorer… » répondrez-vous, penaud.

Pourquoi CSS est-il si pénible à utiliser ?

Dans un essai, Bert Bos, l'un des inventeurs de CSS, livre une explication aux limites syntaxiques du langage (http://bkaprt.com/sass/3/) :

> *CSS n'inclut aucune des fonctions plus puissantes que l'on retrouve dans d'autres langages de programmation : les macros, les variables, les constantes symboliques, les opérateurs conditionnels, les expressions employant des variables, etc. La raison en est que toutes ces choses donnent certes plus de marge de manœuvre aux utilisateurs plus expérimentés, mais que les novices risquent de s'y perdre, voire d'être si intimidés qu'ils ne toucheront même pas à CSS. Il s'agit d'établir un équilibre. Et pour CSS, l'équilibre est différent d'autres langages de programmation.*

Les premiers architectes de CSS étaient soucieux de son adoption. Ils souhaitaient (à juste titre) permettre au plus grand nombre de créer des sites web. Ils voulaient que CSS soit assez puissant pour styler des pages web et séparer le contenu de la présentation tout en étant facile à comprendre et à utiliser. Je n'y vois aucun mal, mais nous avons néanmoins un travail à faire, et ce travail devient de plus en plus compliqué, plus nuancé et plus difficile à entretenir et à rendre « future-proof ».

Heureusement, plusieurs options peuvent nous venir en aide, et Sass est l'une d'entre elles.

QU'EST-CE QUE SASS ?

Sass est un préprocesseur CSS - une couche intermédiaire entre la feuille de styles que vous composez et le fichier `.css` qu'obtient le navigateur. Sass (pour Syntactically Awesome Stylesheets) comble les lacunes de CSS en tant que langage de programmation et permet de rédiger du code DRY plus rapide, plus efficace et plus simple à entretenir (**FIG 1**).

Le site web de Sass (http://bkaprt.com/sass/4/) le décrit succinctement ainsi :

> *Sass est un métalangage qui s'ajoute à CSS et permet de décrire le style d'un document de manière propre et structurelle, avec plus de puissance que ne le permet une feuille de styles plate. Sass offre à la fois une syntaxe plus simple et plus élégante et implémente diverses fonctionnalités permettant de créer des feuilles de styles plus faciles à gérer.*

Alors que le CSS normal ne permet toujours pas d'utiliser des variables, des mixins (des blocs de style réutilisables) et autres réjouissances, la syntaxe de Sass permet tout cela et bien plus encore. Sass traduit (ou compile) ensuite cette syntaxe pour former de bons vieux fichiers CSS par le biais d'un programme en ligne de commande ou d'un plug-in de framework web.

Pour être plus précis, Sass est une extension de CSS3 et sa syntaxe SCSS (« Sassy CSS ») - dont nous parlerons dans un instant - est un surensemble de CSS3, ce qui signifie que tout document CSS3 valide est également un document SCSS valide. Ce point est essentiel, car il permet un passage à Sass « en douceur ». L'apprentissage de la syntaxe de Sass se fait sans effort, et vous pourrez l'utiliser dans la proportion qui vous convient. Cela signifie également que vous pourrez convertir vos feuilles de styles du CSS en SCSS en plusieurs étapes, à mesure que vous découvrirez les fonctionnalités de Sass.

Par la suite, lorsque vous vous serez familiarisé avec Sass (ce qui ne devrait pas prendre beaucoup de temps), vous aurez vraiment l'impression que c'est une extension naturelle de CSS - comme s'il comblait toutes les lacunes de la spécification CSS. Dès le moment où j'ai commencé à utiliser Sass, cela ne m'a

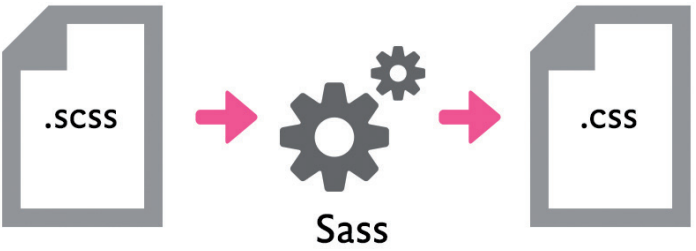

FIG 1 : Sass convertit sa propre « super syntaxe » en CSS ordinaire.

jamais paru difficile ni laborieux : c'était simplement ce que CSS aurait toujours dû être. Quand vous l'aurez essayé, vous ne pourrez plus vous en passer.

Par ailleurs, Sass participe à l'amélioration de CSS. En permettant de tester certaines fonctionnalités qui ne sont pour le moment possibles qu'avec l'aide d'un préprocesseur, il offre aux créateurs de CSS la possibilité d'implémenter des fonctionnalités expérimentales. Il n'est pas impossible qu'à terme, certaines fonctionnalités de Sass soient incorporées dans la spécification de CSS.

SYNTAXE DE SASS

Il existe de fait deux syntaxes différentes dans Sass. Je viens de mentionner SCSS, la plus récente. Les fichiers SCSS prennent une extension de fichier en `.scss`. C'est la syntaxe que je préfère et que je recommande d'utiliser pour les raisons suivantes :

- Comme SCSS est un surensemble de CSS3, je peux rédiger mes feuilles de styles comme je le fais depuis dix ans.
- Il est facile de convertir progressivement des feuilles de styles existantes pour utiliser les fonctionnalités de Sass.
- Cela ne change rien au formatage du code.

Un simple exemple de SCSS

Prenons un exemple de syntaxe SCSS. On définit ici une variable qu'on utilise ensuite dans une déclaration CSS.

```
$pink: #ea4c89;

p {
  font-size: 12px;
  color: $pink;
}
p strong {
  text-transform: uppercase;
}
```

Ce qui sera compilé ainsi :

```
p {
  font-size: 12px;
  color: #ea4c89;
}
p strong {
  text-transform: uppercase;
}
```

Ce code devrait vous être familier, à l'exception de la variable `$pink` (nous aborderons les variables dans la suite de ce livre). SCSS vient s'ajouter au CSS que vous savez déjà écrire. C'est pour cette raison que je l'aime autant.

La syntaxe Sass originale avec indentation

La syntaxe originale de Sass, par contre, c'est une autre paire de manches. Certains préféreront son style dépouillé, sans accolades ni points-virgules. Si vous êtes habitué aux langages de programmation « sobres » comme Ruby ou Python, la syntaxe de Sass devrait vous être familière, et peut-être la préférerez-vous.

Voici l'exemple précédent exprimé en syntaxe Sass originale, qui se compilera exactement de la même façon que le code en SCSS.

```
$pink: #ea4c89

p
  font-size: 12px
  color: $pink

p strong
  text-transform: uppercase
```

Finis les accolades et les points-virgules, ne restent plus que les espaces et l'indentation pour définir la structure des déclarations. Certes, c'est plus propre et plus simple, et certains d'entre vous seront peut-être plus enclins utiliser cette syntaxe. La rédaction est plus rapide et le code plus léger. Mais pour les raisons évoquées précédemment, je préfère toutefois SCSS, plus proche de CSS.

Les exemples du chapitre suivant emploieront tous la syntaxe SCSS. Si vous préférez la syntaxe de Sass, la conversion est simple à effectuer. Toutes les fonctionnalités de Sass que nous allons étudier pourront être appliquées avec les deux syntaxes. Ce n'est qu'une question de préférence.

MYTHES SUR SASS

Je vous ai dit que j'avais eu des réticences à essayer Sass, en partie à cause des préjugés que j'avais avant de l'utiliser. Est-ce que je vais devoir apprendre à coder en Ruby et à bidouiller la ligne de commande ? Vais-je devoir changer complètement ma façon de rédiger des feuilles de styles ? Le code CSS produit sera-t-il lourd et illisible ?

Évidemment, la réponse à chacune de ces questions est « non », mais je vois que certains se les posent encore. Alors, dissipons quelques malentendus.

J'ai peur de la ligne de commande !

Je ne suis pas un expert de la ligne de commande, mais j'ai appris quelques bases ici et là au fil des années - suffisamment pour me mettre dans le pétrin. Je n'ai pas peur de parcourir un

système de fichiers ou d'utiliser Git en ligne de commande, par exemple.

Cela dit, je compatis avec les designers et les développeurs front-end qui ne veulent pas s'y aventurer. Certaines personnes ont une phobie de la ligne de commande. Pour Sass, il n'y a pas grand-chose à faire en ligne de commande – en fait, il n'y a qu'une seule commande à connaître. De plus, il existe des applications et des frameworks web qui permettent de s'en passer complètement. J'en parlerai au chapitre suivant.

Alors si vous êtes vous-même phobique de la ligne de commande, ne vous découragez pas avant d'avoir essayé Sass !

Je ne veux pas changer ma façon d'écrire en CSS !

C'est de ce mythe dont j'ai le plus souffert. J'ai une manière bien particulière de créer et d'organiser mes feuilles de styles. C'est un véritable travail d'artisan. Mais comme la syntaxe SCSS est un surensemble de CSS3, je n'ai rien eu à changer à ma façon d'écrire. Les commentaires, la présence ou l'absence d'indentation, toutes vos préférences de formatage peuvent rester identiques lorsque vous travaillez sur des fichiers `.scss`. Quand j'ai compris ça, je me suis jeté à l'eau sans crainte.

Je ne veux pas que Sass change ma façon de concevoir !

Le revers de la médaille, c'est que Sass ne résoudra pas tous vos problèmes et ne guérira pas vos mauvaises habitudes. Des feuilles de styles lourdes et inefficaces seront tout aussi lourdes et inefficaces avec Sass. Vous devrez toujours faire preuve d'organisation et de réflexion. Dans certaines situations, Sass risque même d'aggraver les mauvaises pratiques, comme nous le verrons par la suite. Mais lorsqu'il est utilisé à bon escient, Sass peut être un outil précieux pour la création de sites Web.

Maintenant que nous avons évacué toutes les inquiétudes, commençons à nous amuser un peu. Vous serez ébahi par les possibilités de Sass. Dans le chapitre suivant, nous allons établir notre workflow – comment intégrer Sass dans votre méthode de travail et utiliser la ligne de commande ou une application pour compiler les fichiers CSS. Alors, partons à la Sass !